#15

IN THE UNITED STATES PATENT AND TRADEMARK OFFICE

Reissue Application No. 90/694,416

Reexamination Control No. **90/005,733**; and

Reexamination Control No. 90/005,776

Inventors: Collins et al.

Patent No. 5,848,159

Issued: December 8, 1998

For: **PUBLIC KEY CRYPTOGRAPHIC APPARATUS AND METHOD**

CERTIFICATE OF MAILING

I hereby certify that this paper (along with any paper referred to as being attached or enclosed) is being deposited with the United States Postal Service as First Class Mail addressed to: Assistant Commissioner for Patents, Box: Reexam, Washington, DC, 20231 on 4/9, 2001

By: D'Evelyn THoran

Attorney Docket No. 20206-125 (PT-TA410)
Attorney Docket No. 20206-126 (PT-TA410US-4)
Attorney Docket No. 20206-127 (PT-TA410US-5), respectively.

Assistant Commissioner for Patents
Box: Reexam
Washington, D.C. 20231

SUPPLEMENTAL INFORMATION DISCLOSURE STATEMENT

Submitted herewith are references listed on the attached form PTO-1449 of which Patentees and Reissue Applicants are aware and believe to be material to the aforementioned Reexaminations of the U.S. Patent and examination of the above-referenced Reissue Application, and in respect of which there may be a duty to disclose in accordance with 37 CFR 1.56.

The filing of this information disclosure statement shall not be construed as a representation that a search has been made (37 CFR 1.97(g)), nor as an admission that the information cited is, or is considered to be, material to patentability, nor an admission that no other material information exists.

The filing of this information disclosure statement shall not be construed as an admission against interest in any manner. Notice of January 9, 1992, 1135 O.G. 13-25, at 25.

Respectfully submitted,

DATE: 4/7/, 2001

Leah Sherry
Reg. No: 43,918

OPPENHEIMER WOLFF & DONNELLY LLP
Customer No. **25696**
1400 Page Mill Road
Palo Alto, CA 94304
Tel: (650) 320-4000
Fax: (650) 320-4100

#15

PE
APR 11 2001
PATENT & TRADEMARK OFFICE

FORM PTO-1449 U.S. DEPARTMENT OF COMMERCE, PATENT AND TRADEMARK OFFICE	REISSUE APPLICATION NO. **09/694,416** REEXAMINATION CONTROL NO. **90/005/733** REEXAMINATION CONTROL NO. **90/005/733** Orig. PATENT NO. **5,848,159**	ATTY DOCKET NO.: 20206-125 (PT-TA410) 20206-126 (PT-TA410US-4) 20206-127 (PT-TA410US-5), respectively.
INFORMATION DISCLOSURE STATEMENT BY APPLICANT	INVENTORS COLLINS et al.	
	ISSUE DATE December 8, 1998	GROUP

U. S. PATENT DOCUMENTS

EXAMINER INITIAL		DOCUMENT NUMBER	DATE	NAME	CLASS	SUBCLASS	FILING DATE IF APPROPRIATE
	AA	4,514,592	4/30/1985	Miyaguchi	178	22.11; 22.14	7/14/1982
	AB	5,046,094	9/3/1991	Kawamura	380	46; 28	2/2/1990
	AC	5,343,527	8/30/1994	Moore	380	4; 25; 30	10/27/1993

FOREIGN PATENT DOCUMENTS

		DOCUMENT NUMBER	DATE	COUNTRY	NAME	CLASS	SUBCLASS	TRANSLATION YES NO
	AD							

OTHER DOCUMENTS (Including Author, Title, Date, Pertinent Pages, Etc.)

	AE	P. J. Flinn et al. Using the RSA Algorithm for Encryption and Digital Signatures: Can you Encrypt, Decrypt, Sign and Verify without Infringing the RSA Patent?" July 9, 1997, Alston & Bird LLP, http://www.cyberlaw.com/rsa.html

EXAMINER	DATE CONSIDERED

EXAMINER: Initial if citation considered, whether or not citation is in conformance with MPEP 609; draw line through citation if not in conformance and not considered. Include copy of this form with next communication to applicant.

Using the RSA Algorithm for Encryption and Digital Signatures:

Can You Encrypt, Decrypt, Sign and Verify without Infringing the RSA Patent?

Patrick J. Flinn and James M. Jordan III[*]

(c) 1997 Alston & Bird LLP

July 9, 1997

"Public key cryptography," a method for encrypting messages to be transmitted over an insecure channel, and "digital signatures," a method for authenticating the author of a message transmitted over an insecure channel, are emerging as fundamental tools for conducting business securely over the Internet. These technologies are widely expected to be used to conduct billions of dollars in electronic commerce within the next few years. However, the broad deployment of these technologies is substantially burdened by licensing demands being made by the owner of United States Patent No. 4,405,829, which is known as the "RSA Patent." It has become commonly accepted Internet lore that the RSA Patent covers most of the commonly used techniques for public key encryption and digital signatures, and that a patent license from the owner of the RSA Patent is necessary to employ these techniques. As this article explores in some detail, however, the RSA Patent is far more limited in scope and far more vulnerable to a validity challenge than is generally assumed.

The RSA Algorithm and the RSA Patent

The RSA Algorithm was named after Ronald Rivest, Adi Shamir and Leonard Adelman, who first published the algorithm in April, 1977.[1] Since that time, the algorithm has been employed in the most widely-used Internet electronic communications encryption program, Pretty Good Privacy (PGP).[2] It is also employed in both the Netscape Navigator and Microsoft Explorer web browsing programs in their implementations of the Secure Sockets Layer (SSL), and by Mastercard and VISA in the Secure Electronic Transactions (SET) protocol for credit card transactions.

The RSA Algorithm is claimed in the RSA Patent, which was issued to Drs. Rivest, Shamir and Adelman, who exclusively licensed the patent nine days later to RSA Data Security, Inc., a company which was originally controlled by the inventors but is now a wholly-owned subsidiary of a Boston based company called Security Dynamics Technology, Inc. RSA Data has to date filed three lawsuits alleging infringement of the RSA Patent. Two were settled prior to trial, and the third is still pending. Other litigation threats have been made regarding alleged infringements of the patent, including threats against non-commercial implementations for use by the Internet community. The patent expires on September 20, 2000, but that will be enough time for the patent to have a profound impact on the development of electronic commerce.

The existence of the patent, and RSA Data's aggressive litigation posture, have chilled the interest in both commercial and non-commercial implementations of public key encryption and digital signature technologies. Many have taken for granted the bald assertion that the "RSA Algorithm is patented," without examining the patent itself, or more particularly, the claims of the patent.[3] As we set forth in this article, however, a careful review of the patent reveals that the patent is not necessarily as broad as publicly asserted. More particularly, the decryption operation, standing alone, is not independently claimed at all in the patent. These weaknesses in the patent may be particularly relevant for digital signature operations because they may allow a developer to implement the protocol for verifying an RSA-generated digital signature without infringing the patent. In addition, if one separates the generation of the key pairs from the encryption operation, the claims of the patent do not cover the encryption (or signing) function by itself.

Basic Uses of Public Key Encryption and Digital Signatures

The RSA Algorithm is only one implementation of the more general concept of public key cryptography, which permits two parties who have never met and who can only communicate on an insecure channel to nonetheless send secure and verifiable messages to each other. The Internet as currently structured is an insecure communications channel with an obvious use for such technologies. Indeed, the greatest expected growth for public key techniques is in Internet-related communications.

With public key techniques, each user has two different keys, one made available to the public and the other kept secret.[4] One of the keys is used to encrypt a message, and the other is used to decrypt the message. If Alice wants to send a secret message to Bob, for example, she looks up Bob's public key and uses it to encrypt the message. Because Bob's *public* key cannot undo the encryption process, no one who intercepts the message can read it. Only Bob, who possesses the secret key corresponding to his public key, can read the message. Alice never has to meet Bob out of the hearing of others to exchange keys or passwords; this is a substantial improvement over older encryption methods in which an exchange of private keys was necessary.

This system can also be used as a means for Bob to be sure a message comes from Alice. If Alice wants to sign a message, she can encrypt it with her private key.[5] When Bob receives an encrypted message which purports to be from Alice, he can obtain Alice's public key and decrypt the message. If a readable message emerges, Bob can have confidence that the message came from Alice, because Alice's public key would only properly unlock a message which was locked with her private key (known only to Alice).[6]

Of course, digitally signing the message does not make the content of the message private, because anyone with Alice's public key can read a message she encrypted with her private key. Alice can send a private, signed message to Bob, however, by first encrypting the message with Bob's public key (so only Bob can read it with his private key) and then encrypting the message a second time with her private key, forming her signature. Anyone who receives the message can use Alice's public key to undo the second encryption, but only Bob (or someone with Bob's private key) can undo the first encryption step and actually read the message. All of these complex-sounding manipulations can be made quite manageable with well-written software.

The RSA Implementation of Public Key Encryption and Digital Signatures

The Arithmetic in the RSA System

Typical encryption techniques use mathematical operations to transform a message (represented as a number or a series of numbers) into a *ciphertext*. Mathematical operations called *one way functions* are particularly suited to this task. A one way function is one which is comparatively easy to do in one direction but much harder to do in reverse. As a trivial example, it is comparatively easy to square a two digit number; with a little concentration, many people can probably multiply 24 by 24 without using a pencil and paper. One the other hand, calculating the square root of the number 576 is much

harder, even with a pencil and paper.

The RSA system uses one way functions of a more complex nature. Specifically, the system uses *modular arithmetic* to transform a message (or pieces of the message, one piece at a time) into unreadable ciphertext. Modular arithmetic is often called "clock" arithmetic, because addition, subtraction, and the like, work like telling time. In a 12-hour system, four hours after 10:00 is not 14:00 (10 + 4 is not equal to 14); it is 2:00. This is because we subtract out 12 (or any multiples of 12) after doing the addition. In modular arithmetic notation, the operation might look like this:

$$2 = (10+4) \bmod 12$$

$$2 = 14 \bmod 12$$

One can do multiplication in modular arithmetic much the same way addition is done in the above example:

$$2 = (7*2) \bmod 12$$

$$2 = 14 \bmod 12$$

This process is sometimes called *modular reduction*. By subtracting out the modulus (and all multiples of the modulus) a number is "reduced" to a much smaller number. When the number 14 is "reduced" to the number 2 in the above example, one can say that "14 is reduced modulo 12."

The RSA system uses multiplication in modular arithmetic. Instead of multiplying one number by a different number (as (7) is multiplied by (2) in the above example), The RSA system multiplies one number (called the *base*) by itself a number of times. The number of times a base is multiplied by itself is called the *exponent*:

$$16 = 2*2*2*2$$

$$16 = 2^4$$

In this example, the number (2) is the base, and is multiplied by itself four times, making the exponent the number (4).

In the RSA encryption formula, the message (represented by a number M) is multiplied by itself (e) times (called "raising (M) to the power (e)"), and the product is then divided by a modulus (n), leaving the remainder as a ciphertext (C):[7]

$$C = M^e \bmod n$$

This is a hard operation to undo -- when (n) is very large (200 digits or so) -- even the fastest computers using the fastest known methods could not feasibly recover the message (M) simply from knowing the ciphertext (C) and the key used to create the message ((e) and (n)).

In the decryption operation, a different exponent, (d) is used to convert the ciphertext back into the plain text:

$$C = M^d \bmod n$$

The modulus (n) is a composite number, constructed by multiplying two prime numbers,[8] (p) and (q), together:

$$n = p * q$$

The encryption and decryption exponents, (d) and (e), are related to each other and the modulus (n) in the following way:[9]

$$d = e^{-1} \bmod ((p-1)(q-1))$$[10]

To calculate the decryption key, one must know the numbers (p) and (q) (called the factors) used to calculate the modulus (n). When (n) is a sufficiently large number, it is infeasible, using known algorithms and the fastest computing techniques, to calculate the prime number factors of (n).

The RSA Algorithm may be divided, then, into three steps:

(1) *key generation*: in which the factors of the modulus (n) (the prime numbers (p) and (q)) are chosen and multiplied together to form (n), an encryption exponent (e) is chosen, and the decryption exponent (d) is calculated using (e), (p), and (q).

(2) *encryption*: in which the message (M) is raised to the power (e), and then reduced modulo (n).

(3) *decryption*: in which the ciphertext (C) is raised to the power (d), and then reduced modulo (n).

Using the RSA Algorithm for Privacy and Digital Signatures

When the RSA Algorithm is used in a public key system, the modulus (n) and one of the exponents (arbitrarily, we can assume (e)) are published. The other exponent (d) is kept secret, as are (p) and (q), the factors of (n). Each user holds his or her own keys, and knows the public key of the other user or users. Alice, for example, knows her own public key (e_{alice} and n_{alice}), her own private key (d_{alice}), and Bob's public key (e_{bob} and n_{bob}). Bob knows the converse: his public key (e_{bob} and n_{bob}), his private key (d_{bob}) and Alice's oublic key (e_{alice} and n_{alice}).

For Alice to send Bob a private message only Bob can read, she performs the following operation on the message (M):

$$C = M^{ebob} \bmod n_{bob}$$

Bob, who is the only one to possess his private key (d_{bob}), performs the following to recover the message (M):

$$M = C^{dbob} \bmod n_{bob}$$

To sign the message, Alice encrypts with her own private key:

$$C = M^{dalice} \bmod n_{alice}$$

Because only Alice possesses d_{alice}, only she can create this ciphertext C. Anyone in possession of her public key (e_{alice} and n_{alice}) can verify the signature, however:

$$M = C^{ealice} \bmod n_{alice}$$

It bears note that (p) and (q), the factors of (n), are not needed for encryption or decryption; they are only used in the key generation step (creating the modulus (n) and the second exponent). In addition, while it is important for key generation purposes that the modulus (n) be the product of two

prime numbers, the exponentiation and modular arithmetic operation would work just as well with prime numbers (which are by definition evenly divisible only by themselves and the number 1).

The RSA Patent

Anyone who "makes, uses, offers to sell or sells" a patented invention without the permission of the patent owner can be liable for patent infringement.[11] The boundaries of the patent are defined in the claims portion of the patent.[12] Accordingly, in order to determine whether a particular product, method or process infringes a patent, one must start with the text of the claims themselves.

There are 40 claims in the RSA Patent, but only ten of them are *independent* claims.[13] Independent claims are claims which do not incorporate other claims by reference. To infringe a dependent claim, one must first infringe the independent claim (or claims) incorporated by reference in the dependent claim. Conversely, if one does not infringe the independent claim(s) incorporated by the dependent claim, by definition one does not infringe the dependent claim. Accordingly, a review of the independent claims in the RSA Patent is sufficient for purposes of this discussion. As we will also see, a detailed review of the broadest independent claim in the RSA Patent (Claim 23) will lead without much further ado to a logical conclusion about the other nine independent claims, and in turn to a conclusion about all the claims in the RSA Patent: that none of these claims are infringed by performing a typical digital signature verification.

The claim with the least number of elements (and thus the broadest claim in the patent) is Claim 23, which provides:

23. A method for establishing cryptographic communications comprising the step of:

encoding a digital message word signal M to a ciphertext word signal C, where M corresponds to a number representative of a message and

$$0 <= M <= n-1$$

where n is a composite number of the form

$$n=p*q$$

where p and q are prime numbers, and

where C is a number representative of an encoded form of message word M,

wherein said encoding step comprises the step of:

transforming said message word signal M to said ciphertext word signal C whereby

$$C \text{ [is congruent to] } M^e \pmod{n}$$

where e is a number relatively prime to (p-1)*(q-1).

Accordingly, the elements of this claim require an accused infringer to perform the following steps: [14]

- Establish cryptographic communications;
- Ensure that the message (M) is greater than or equal to zero and less than or equal to (n-1);
- Define the modulus (n) by selecting prime numbers (p) and (q) and multiplying them together;

- Define the encryption exponent (e) such that it is relatively prime[15] to (p-1)*(q-1); and

- Encode message (M) into ciphertext (C) by raising (M) to the power of (e) and
- then reducing modulo (n).

Does Claim 23 Cover Signature Verification?

As noted above, to verify an RSA-generated digital signature, the recipient takes the ciphertext transmitted to him and decrypts the ciphertext with the sender's public key. If the message decrypts properly, the signature is genuine. Importantly, two of the three fundamental steps in the RSA system are not performed in the signature verification step: *key generation*, where the parameters of the modulus (n) and the exponents (e) and (d) are set; and *encryption*, where the message (M) is raised to the power of (e) and then reduced by the modular operation. Only the third, *decryption* step is performed. The keys are generated by the sender (signer) and the encryption step has already been completed in the signing step.

Accordingly, a person who merely *verifies* an RSA signature arguably does none of the steps contained in Claim 23, and certainly does not do them all. Most significantly, the decryption operation plainly does not constitute "encoding a digital message word signal" into "ciphertext." The verification operation transforms "ciphertext" into a "message word signal," not the reverse. The patent makes clear, moreover, what constitutes a "message" and what constitutes "ciphertext," and how "decoding" and "encoding" are different.[16] These terms are not interchangeable.[17]

The RSA signature verification steps of transforming ciphertext C into a message M using the decryption exponent (d) are separately claimed in *dependent* claim 24:

24. The method according to claim 23 comprising the further step of:

decoding said ciphertext word signal C to said message word signal M,

wherein said decoding step comprises the step of:

transforming said ciphertext word signal C, whereby:

$$M \text{ [is congruent to] } C^d \pmod{n}$$

where d is a multiplicative inverse of e (mod (lcm ((p-1), (q-1)))).

Because Claim 24 incorporates all of the elements of Claim 23 by reference, one cannot infringe Claim 24 simply by performing the decryption steps alone.

Do the Other RSA Patent Independent Claims Cover Signature Verification?

The analysis of Claim 23 above should apply with equal force to the other independent claims. Claims 1, 13, and 18 require key generation, encoding, and de-coding. Claims 3, 8, 25, 29 do not contain the decoding step (subsequent dependent claims add that step), but have at least the elements of Claim 23, plus others. Claims 33 and 37 claim a special case of the use of the RSA method, and thus are also more limited than Claim 23. Thus, under this analysis none of the independent claims of the RSA Patent (and therefore none of the dependent claims) are infringed by performing a typical digital signature verification.

Does Claim 23 Cover Generation of a Digital Signature?

It appears that the process of *generating* an RSA signature also may be done without infringing Claim

23. To create an RSA digital signature, the sender encrypts the message M with her private key, and transmits it to the receiver. The encoding step is clearly claimed in Claim 23 (and other independent claims), and thus the argument stated above regarding verification (decryption) is not available. However, Claim 23 also requires key generation, and that step need not be performed by software creating a digital signature if the keys are already supplied. (All of the other independent claims require the generation of the keys meeting the RSA Algorithm parameters.)

The step of generating the numbers comprising RSA keys (p, q, n, e and d) can easily be separated from the encryption step. The same keys can be used over and over again for multiple signatures; indeed, they can be used for as long as one has confidence that the keys have not been compromised. Moreover, many RSA-licensed products generate keys which can be separated from the software which generated them. Thus, as a practical matter, it should be possible to use keys generated by licensed RSA software in order to create digital signatures with other, unlicensed software.

The owner of the RSA Patent would have difficulty arguing successfully that the encryption operation itself is covered by the patent, separate from the generation of the keys. This is because the concept of using exponents in modular arithmetic for encryption was invented and disclosed before the RSA system was invented. In 1975, two years before the RSA method was invented, Martin Hellman and Stephen Pohlig at Stanford University invented the Pohlig-Hellman encryption system,[18] which is identical to the RSA method, except that the modulus is a prime number, as opposed to the product of two primes:

Comparison of RSA and Pohlig-Hellman		
	RSA System	*Pohlig-Hellman*
Encryption Operation	$C = M^e \bmod n$	$C = M^e \bmod p$
Decryption Operation	$M = C^d \bmod n$	$M = C^d \bmod p$
modulus	p * q (prime numbers)	p (prime number)
Encryption exponent (e)	e relatively prime to (p-1)*(q-1)	e relatively prime to (p-1)
Decryption exponent (d)	$d = e^{-1} \bmod ((p-1)*(q-1))$	$d = e^{-1} \bmod (p-1)$

The exponentiation and modular reduction steps in RSA and Pohlig-Hellman will work exactly the same regardless of whether the modulus is a prime number or the product of two primes. Once the modulus and the exponents are defined, the mathematical operations for encryption and decryption are *identical* in both the Pohlig-Hellman and RSA systems. A software module written to perform Pohlig-Hellman encryption or decryption would work just as well using a modulus and exponents generated with an RSA system. Accordingly, even if the inventors had attempted to claim the encryption step alone, without regard to key generation, the prior Pohlig-Hellman invention would have prevented such a claim from being valid.[19]

What About Contributory Infringement?

The discussion above relates to direct infringement of RSA Patent -- whether one performs all of the steps one of the claims -- but does not consider all of the possible ways in which one can be liable for patent infringement. Patent law also makes liable someone who "actively induces" infringement of a patent, or someone who contributes to the infringement by another if he "offers to sell or sells within the United States . . . a component of a patented machine, manufacture, combination or composition, or a material or apparatus for use in practicing a patented process, constituting a material part of the invention, knowing the same to be especially made or especially adapted for use in an infringement of such patent, and not a staple article or commodity of commerce suitable for substantial noninfringing use . . . ."[20] The owner of the right to enforce the RSA Patent could attempt to attack those who sign or verify as contributory infringers, and those who develop the software which enables this activity as active inducers of infringement.

The major flaw in such claims is that for there to be either contributory infringement or inducement to

infringe, there must be direct infringement.[21] One who merely verifies an RSA generated signature simply has not performed all of the steps claimed in the patent. The missing encryption and key generation steps have been performed by another (presumably licensed) user of the technology. Similarly, assuming a signer uses a licensed method for generating the keys, or if the signer does not herself generate or cause to be generated the RSA keys, there is no direct infringement by using an unlicensed method to perform the signing operation.[22] As long as the user only performs the signing or verification steps, there is no direct infringement. With no direct infringement, supplying the means for signing or verification is not contributory infringement.

A closer question of contributory infringement would arise if a developer of signing software knew of the unlicensed generation of RSA keys and deliberately adapted the software to assist in the use of those unlicensed keys in performing signature operations. (This issue is not present for verification; by definition, the person who verifies an RSA signature does not generate, or have access to, the private keys used for signing.) The RSA Patent rights owner could argue that the signing software was "especially adapted" for use in infringing the patent. Then, the argument would run, a user who both generated the keys using unlicensed means *and* the developer's software for signing directly infringed the patent. This would supply the missing direct infringement and expose the developer to contributory infringement liability.

The outcome to this argument would turn in large measure on the nature of the developer's signing software. When an article is accused of being "especially adapted" for use in an infringing process or method, the article as a whole is considered, not just some particular feature or ingredient.[23] If, taken as a whole, the article has substantial, non-infringing uses, the contributory infringement is not present, even if in some modes the article can assist in conduct which otherwise infringes the patent.[24] For example, software adapted to encrypt a message using Pohlig-Hellman keys, for example, would work as easily with RSA Keys. Assuming that Pohlig-Hellman operations were considered "substantial," the use of the identical encryption and decryption operations certainly would be non-infringing.[25] Accordingly, it is quite possible to conceive of software which, while capable of performing RSA encryption, would be found as a whole to have substantial non-infringing uses. If software were developed which, taken as a whole, had substantial non-infringing uses, then the developer should not be found liable for contributory infringement merely because the keys were generated by unlicensed means.

A Real World Example -- Secure Sockets Client

One of the most common uses of public key technology is in the Secure Sockets Layer (SSL) protocol, used by Netscape Navigator and other web browsers for secure communications over the internet. Secure sockets allows the user -- typically an individual working from a personal computer at home (called the *client*) -- to communicate with a web site computer (called the *server*). The server might be operated by a merchant and the user might wish to have the client computer send a credit card number to the server to order goods. The secure sockets protocol uses public key techniques to encrypt the credit card number so that the number is not sent in plaintext over the internet. According to the analysis set forth in this article, the *client* should be able to perform all of the RSA encryption and verifications steps without infringing the RSA Patent.

In simplified form, the secure sockets protocol using the RSA method works as follows.[26] After a client makes context with the web site server, and a secure session is to be established, the server transmits a message the client containing (1) the server's public key; and (2) a digital signature from a certificate authority certifying that the public key the server claims as its own is, in fact, its own. The client next verifies the server's public key using the signature authority's public key which is installed with the browser software on the client's computer.[27] In this way, the client can be assured that the server is who it claims to be. The client is then ready to send confidential information (such as a credit card number) to the server.

The client will encrypt the credit card number, but will not use RSA or another public key technique to actually encrypt the data. RSA is much slower than other conventional encryption methods which use the same key to encrypt and decrypt. While this might not matter if all that is being encrypted is a

single credit card number, in practice a secure session with a web browser may involve the transmission of thousands of bytes of information.

To avoid painfully long delays, the client will generate a random number for use as a conventional, symmetric key to encrypt the all of the data being sent during the secure session. The client then encrypts this key using the server's RSA public key transmitted in the opening message from the server. The server receives this encrypted key, and decrypts it using its secret key. Both the client and the server are now in possession of a shared, symmetric key which both use to encrypt and decrypt all subsequent data being sent during the secure session.

In this protocol, the client does only two public key steps, and does *not* perform any key generation steps. First, the client verifies the server's public key. To perform the verification step, it uses the certificate authority's public key. That key, necessarily generated by the certificate authority, can be assumed reasonably to be generated by a licensed entity.[28] The second step -- encrypting the symmetric encryption key for the session -- uses the public key supplied by the server. That key also can be presumed to have been generated by a licensed method. Although both decryption (verification) and encryption (of the session key) steps are performed by the client, there is every reason to believe that the key generation steps are performed by licensed methods.

Accordingly, because the client software does not generate any keys, the client does not directly infringe the any of the claims of the RSA patent. Although the client does use two public keys (the server's and the certificate authority) for encryption and decryption, those keys are almost certainly generated licensed means. Accordingly, the client does not contribute to, or induce the infringement of, the RSA Patent.

Some Questions About the Validity of the RSA Patent Claims

The analysis in this article thus far has assumed that the broad claims of the RSA Patent, and particularly Claim 23, are valid. While patents are presumed valid, and an infringer has the burden of proving invalidity by clear and convincing evidence, some vulnerabilities of the RSA Patent have been exposed in the litigation RSA Data has been involved in. Among other weaknesses, the following appear: (1) some of the RSA claims may not cover patentable subject matter, (2) the inventors appear not to have disclosed in the specification the "best mode" of implementing the invention, as required by the patent laws, and (3) a real question exists regarding whether the invention is obvious in light of the Pohlig-Hellman work. Each of these weaknesses will be considered below.

How Can You Patent an Algorithm?

Those with passing familiarity with patent law may be startled at the assertion that a cryptographic algorithm can be claimed in a patent. The United States Supreme Court ruled in 1972 that an algorithm, which it defined as a "procedure for solving a given type of mathematical problem," was not a "process, machine, manufacture, or composition of matter" within the meaning of section 101 of the Patent Act,[29] and thus was not patentable subject matter.[30] Six years later, the Court reaffirmed this rule, holding that even if the applicant wanted to limit the claim to use of the algorithm in a specific application, it was still not within the allowable subject matter of section 101. [31] Cryptographic algorithms would seem to fall squarely within this proscription against patenting algorithms.

In 1981, however, a breakthrough Supreme Court decision paved the way for patent claims containing algorithms. The case, *Diamond v. Dehr*,[32] involved an improved process for making rubber. The improvement centered on an algorithm used to treat the rubber at specified temperatures. The Court held that when an algorithm is part of an otherwise patentable process (which manufacturing rubber certainly was), the presence of the algorithm among the other elements of the claim did not push the claim outside the bounds of section 101.

While the Supreme Court was deciding these cases, a trio of appellate court decisions refined the rules regarding when and how an algorithm may be incorporated into a patent. The three cases, *In re*

Freeman,[33] *In re Walter*,[34] and *In re Abele*,[35] establish what the Federal Circuit refers to as the *Freeman-Walter-Abele* test for patentability when an algorithm is implicated in a patent claim. The test is stated as follows:

It is first determined whether a mathematical algorithm is recited directly or indirectly in the claim. If so, it is next determined whether the claimed invention as a whole is no more than the algorithm itself; that is, whether the claim is directed to a mathematical algorithm that is not applied to or limited by physical elements or process steps. Such claims are nonstatutory. However, when the mathematical algorithm is applied in one or more steps of an otherwise statutory process claim, or one or more elements of an otherwise statutory apparatus claim, the requirements of section 101 are met.

Arrhythmia Research Technology, Inc. v. Corazonix Corp.[36] In other words, the fact that an algorithm is one of the elements of a claim of a patent does not make the claim invalid. If the algorithm is used as part of a physical process (such as the manufacture of rubber, as in *Dehr*,) or is part of a physical device (such as an electrocardiograph device, as in *Arrhythmia*), the invention is patentable subject matter.

Many of the existing cryptography patents contain attempts to meet this test by reciting the cryptographic algorithm as an element within a physical device. In the RSA Patent, this approach is stretched near if not beyond the breaking point. The only elements of the physical device claim which are *not* descriptions of the algorithm are "a communications channel," "an encoding means," and a "decoding means."[37] The broadest method (or process) claim of the RSA Patent describes using the algorithm to "encode" a "message word signal."[38] By using the most generic references to a device ("encoding means"), and the most generic reference possible to a physical process (encoding a "message word signal"), this type of patent comes as close as possible to claiming the algorithm by claiming the use of the algorithm in essentially all possible machines or with all possible processes. Indeed, it is difficult to imagine how one could use the algorithm without a physical device which could be characterized as an "encoding means," or how one could apply the algorithm in a way which did not "encode" a "signal."

Three recent Federal Circuit cases give insight into how the courts might apply the *Freeman-Walter-Abele* test to a cryptographic algorithm. First, in *In re Alappat*,[39] the Federal Circuit used that test to find valid a patent which claimed an algorithm for displaying a smooth waveform from digital data. "Although many, or arguably even all, of the means elements recited in [the disputed claim] represent circuitry elements that perform mathematical calculations, which is essentially true of all digital electrical circuits," the Federal Circuit noted, the patent was nonetheless proper because "the claimed invention as a whole is directed to a combination of interrelated elements which combine to form a machine for converting discrete waveform data samples into . . . data to be displayed on a display means."[40] *Alappat* can be read to stand for the proposition that, if the physical machine which performs the algorithm is a computer, and if the output from the algorithm is otherwise displayed, it may well be that the "physical device," or "physical process" requirements of the *Freeman-Walter-Abele* test are met.

Another Federal Circuit case, *In re Warmerdam*,[41] further confirms that court's generous view of algorithm-based patents. In *Warmerdam*, the court found patentable an invention claiming a "bubble hierarchy" algorithm used by computer-operated robots to avoid obstructions. Because a claim covered a "machine" (the computer and memory controlling the robot) it was patentable, even though the novelty of the invention consisted solely of the use of the algorithm.[42]

The Federal Circuit has, however, limited the patentability of algorithm claims in at least one recent case. In *In re Schrader*,[43] the court found unpatentable a claim to an invention for processing auction bids. The court distinguished *Abele* and *Arrhythmia* based on the nature of the *input* to the algorithm. In *Abele*, the input was data from an X-ray CAT scan; in *Arrhythmia*, the input was data from a electrocardiograph. Both sources of input data, the Federal Circuit reasoned, involved "subject matter representative of or constituting *physical activity or objects*."[44] Bid data from auction bidders was mere "data gathering," according to the *Schrader* court, and thus was materially different

from X-ray or heart-rate data.[45] The fact that the patent specification discussed displaying the resulting data on display screens did not affect patentability, nor did a claim element for entering the bid data in a "record."[46]

One possible way to apply these cases to a cryptographic algorithm patent is to ask this question: is the input to a cryptographic system more like the data from auctioneers, or is it more like electro-cardiograph, CAT-scan, or rubber temperature data? The latter are representations of a physical object -- a part of the body. Bid data, in contrast, seems much closer to the plaintext message input of a cryptographic system. Both are merely abstract messages intended to be communicated from one party to another. The bid data algorithm of *Schrader* simply transformed the message in a way to extract certain information to the auctioneer. An encryption algorithm simply transforms the message to protect privacy or security.

If the test is, indeed, whether the input to the algorithm is data descriptive of a physical object existing in the real world (such as a heartbeat or the temperature of rubber) as opposed to an abstract message composed by a human being, the validity of a broad cryptography patent such as the RSA Patent is subject to genuine question.

It bears emphasis that the patentability of the claims in the RSA Patent will vary, claim by claim. The attorneys who drafted the claims may have anticipated this problem; they added dependent claims which recite "register means" for accomplishing the steps claimed in the algorithm.[47] (Other dependent and independent claims simply add additional qualifications to the algorithm, and do not add any other physical structures to the claims.) The use of the term "register means" appears to have been intended to encompass a generic computer system at its most basic level. Under the current procedures applied by the Patent Office to patents claiming algorithms, however, one cannot transmute an unpatentable series of algorithm steps into a patentable machine merely by claiming the algorithm along with a generic computer performing the steps.[48] Thus, the addition of the "register means" elements to the claims did not materially improve their validity, and may in fact have limited the scope of the claims.[49] As written, all of the RSA Patent claims consist solely of algorithms combined with the most minimum, generic hardware means possible. A genuine question remains whether, in that form, they claim patentable subject matter.

It is true that patents are presumed valid, and that clear and convincing evidence[50] of invalidity is required to attack an issued patent. This rule exists largely out of deference to the expertise of the Patent and Trademark Office. The question of patentable subject matter, however, has been treated as a question of law; the courts decide such questions *de novo* without deference to any administrative body.[51] Moreover, the RSA Patent was issued at a time when the law on algorithm-based claims was unclear. Indeed, under the current examination guidelines, it is doubtful whether any of the RSA Patent claims would be allowed.[52] Accordingly, one could not have confidence in every case that the examiner was aware of the proper legal standard to be applied.

Does the Patent Disclose the Best Mode?

Section 112 of the Patent Act requires an inventor to fulfill certain requirements in making the details of the invention known in the specification of the patent. One of these obligations requires the inventor to "set forth the best mode contemplated by the inventor of carrying out his invention."[53] The best mode requirement "is intended to ensure that a patent applicant plays 'fair and square' with the patent system. It is a requirement that the *quid pro quo* of the patent grant be satisfied. One must not receive the right to exclude others unless at the time of filing he has provided an adequate disclosure of the best mode known to him of carrying out his invention."[54] Whether the best mode requirement has been met is as a question of fact.[55]

It appears that the principal inventor of the RSA Algorithm, Ronald Rivest, believed that certain criteria in the selection of the prime numbers ((p) and (q)) were important.[56] Evidence of his subjective belief comes from papers he wrote both before and after he applied for his patent. In August, 1977, before he applied for the patent, he wrote:

To gain additional protection against sophisticated factoring algorithms, both (p-1) and (q-1) should contain large prime factors and gcd(p-1,q-1) should be small.[57]

In January, 1978, Dr. Rivest published a paper in response to a proposed attack on his system, in which he gave more details about his preferred method of constructing his system.[58] In this later paper, Dr. Rivest noted that the prior paper "makes definite suggestions as to how the prime numbers p and q should be chosen," but that "this note should help make those suggestions less mysterious."[59] This January paper goes on to give more details about the selection of prime numbers, and further provides yet more details regarding the selection of the exponent (e) which Dr. Rivest preferred for additional security.[60]

While Dr. Rivest's *papers* may have discussed his best mode of implementing the invention, the patent disclosure does not. The RSA Patent specification does refer to one of these papers -- the one whose discussion Dr. Rivest later characterized as "mysterious." The best mode requirement cannot be met merely by references to other papers, however. The disclosure must be in the specification of the patent itself.[61] Accordingly, there is good reason to believe the RSA Patent is invalid because the inventors did not comply with the best mode requirement.

Was the RSA Algorithm Obvious?

Section 103 of the Patent Act provides that a patent is invalid "if the differences between the subject matter sought to be patented and the prior art are such that the subject matter as a whole would have been obvious at the time the invention was made to a person having ordinary skill in the art . . . ." Section 102 of the Patent Act defines what is and what is not prior art for the purpose of applying the obviousness test of section 103.[62] Under section 102(a), prior art includes matters "known or used by others in this country, *or* patented *or* described in a printed publication in this or a foreign country, before the invention thereof by the applicant" (emphasis added), and under section 102(g) prior art includes prior inventions by another. The Pohlig-Hellman paper[63] is likely to be held as prior art under sections 102(a) or (g), as a printed publication, as public knowledge, or as a prior invention.

The file wrapper of the RSA Patent reveals that the inventors of the RSA Patent did not disclose the Pohlig-Hellman paper to the Patent Office and that the examiner did not consider whether the use of a non-prime number in lieu of a prime number (the only difference between Pohlig-Hellman and RSA) was obvious. While the presumption of validity is available even where prior art has not been considered during the prosecution of a patent, that presumption is not as strong where the art was not submitted to the examiner. There are a number of references in the prior art, moreover, to using the problem of factoring composite numbers in cryptography, dating back to the 19th century.[64] Accordingly, should the RSA Patent ever become subject to further litigation, it may not survive a validity challenge based on the Pohlig-Hellman work.

Conclusion

Due largely to luck, bluster, and the naiveté of potential competitors, the owner of United States Patent No. 4,405,829 has enjoyed a virtual monopoly on all uses of the RSA Algorithm. However, a careful scrutiny of the RSA Patent claims, and other details of its disclosure and prosecution, reveal both significant limits to the scope of the patent and material questions regarding its validity. While this article is not intended by the authors as legal advice to the reader nor as an invitation or encouragement to infringe the RSA Patent or any other patent, those interested in implementing this technology should take a close look at the patent -- and obtain the advice of a competent attorney familiar with the proposed project -- before accepting at face value the oft-repeated notion that "RSA is patented."

[*] *Mr. Flinn and Mr. Jordan practice with the law firm of Alston & Bird, LLP, located in Atlanta Georgia. Mr. Flinn is a member of the California and Georgia Bars, and Mr. Jordan is a member of the Georgia Bar and is admitted to practice before the United States Patent & Trademark Office. They*

have represented clients in litigation against RSA Data Security, Inc., regarding, among other things, the validity of the RSA Patent. The opinions expressed in this article are personal, however, and do not necessarily reflect the opinions of their firm, or any client of their firm.

This article consists of statements of personal opinion by the authors, and is neither legal advice nor an invitiation to any person to practice any invention claimed in any patent. Because the specific circumstances of any individual or entity will determine the extent of exposure to liability for infringing a patent, reliance on this or any other article of general applicability should not be substituted for legal advice from an attorney familiar with your particular circumstances.

Portions of this article were previously published in the March, 1997 and June, 1997 editions of *Electronic Banking Law and Commerce Report* (Glasser LegalWorks).

[1] Ronald L. Rivest, Adi Shamir, Len Adelman, "On Digital Signatures and Public Key Cryptosystems," MIT Laboratory for Computer Science Technical Memorandum 82 (April 1977).

[2] The latest version of the PGP program for personal use is PGP for Personal Privacy, Version 5.0, which is available in beta form for download at http://www.pgp.com, and will be in final form in late May 1997. This new version offers a choice between the RSA algorithm, on the one hand, and the DSS/Diffie-Hellman algorithms, on the other.

[3] For example, Bruce Schneier's otherwise excellent book, APPLIED CRYPTOGRAPHY (2d ed.), John Wiley & Sons (1996) at 474, (hereinafter Schneier) contains this overstatement.

[4] A "key" in these systems is not a physical device, but is a number (akin to a password). The encryption key is used as an input to the mathematical formula which transforms a readable message into apparently meaningless gibberish. The decryption key is the number used as an input to the formula which transforms the apparent gibberish back into the original message.

[5] Alice does not have to encrypt the whole message just to sign it. In fact, because such large numbers are used in real-world implementations of this technology, the encryption operation can be quite slow. Instead, Alice can use something called a "hashing algorithm" to transform a message of any size in to a string of numbers of a fixed (and usually smaller) size. If the hashing algorithm is designed properly (and there are a number in current use), different messages should not produce the same hash. Accordingly, the hash acts as a digital "fingerprint" of the message. Alice can then encrypt the hash, not the message itself, with her private key. To verify the signature, Bob simply re-hashes the message using the same hashing algorithm Alice used, and then decrypts Alice's hash with her public key. If the hashes match, the message is authentic and has not been altered in transit.

[6] Bob should have some way of being confident that the number he thinks is Alice's public key is, in fact, her public key. There are a number of alternative proposals and methods for allowing for the verification of public keys, and the debate over them is lively, but the details the competing systems are beyond the scope of this discussion.

[7] Substituting numbers for variables, a modular arithmetic operation looks like this: $1 = 2^4 \bmod 5$ The base (2) multiplied by itself four times (the exponent (4) is equal to 16 (2*2*2*2=16)

$1 = 16 \bmod 5$

When there are 3 five's in 16 (3 times 5 is 15) and a remainder of 1.

[8] A prime number is one which is divisible only by itself and the number 1. The numbers 17 and 31 are prime, for example, but 15 (divisible by 1, 3, 5 and 15) and 21 (divisible by 1, 3, 7 and 21) are not.

[9] The number representing the message (M) should be less than the modulus (n) if decryption is to

succeed. In addition, the multiplicative inverse (d) only exists if (e) is relatively prime to (has no prime number factors in common with) the number (p-1)*(q-1).

[10] The patent defines (d) and (e) as multiplicative inverses modulo the "least common multiple" (or "lcm") of (p-1) and (q-1), a value which divides the value (p-1)*(q-1) used here, resulting in additional possible values for d beyond the simplified version given here. This detail is not important to the discussion in this article, however.

[11] 35 U.S.C. section 271(a).

[12] *E.g., Texas Instruments, Inc. v. U.S. Int'l Trade Commission*, 805 F.2d 1558, 1562 (Fed. Cir. 1986), *reaff'd, reh'g denied*, 846 F.2d 1369 (1988) .

[13] The independent claims are: 1, 3, 8, 13, 18, 23, 25, 29, 33, and 37.

[14] In patent law, as in horseshoes and hand grenades, "almost" counts. One can be liable for patent infringement even if one does not literally infringe all of the elements of a claim. Under a theory called the "doctrine of equivalents," recently upheld by the U.S. Supreme Court, infringement can be found as long as the defendant infringes a "substantial equivalent" of an element of a claim not literally infringed. *Warner Jenkinson Co. v. Hilton Davis Chemical Co.*, 520 U.S. ___, 117 S.Ct. 1352 (March 31, 1997).

[15] A number is "relatively prime" to another number if the two numbers have no factors in common except for the number 1. For example, the numbers 8 and 21 are not prime numbers (each has a factor other than itself and 1) but 8 and 21 are relatively prime to one another because the only common factor they have is the number 1 (the factors of 8 are 1, 2, 4, and 8; the factors of 21 are 1, 3, 7 and 21).

[16] As these terms are used in the RSA Patent the "message" is "encoded" into "ciphertext" and "ciphertext" is sent from the first party to the second. *See* RSA Patent at Col. 1, Line 56- Col.2, Line 10. The recipient receives "ciphertext" and "decodes" the "ciphertext" into the "original message." *Id.* Thus, a person wishing the verify a digital signature must have received "ciphertext," and thus must "decode" it, because the "message" is not transmitted through the communications "channel." *Id.*

[17] Other components in the algorithm probably are interchangeable. For example, the encryption exponent (e) and the decryption exponent (d) each meet the definition of the other. The fact that the exponent (d) is the multiplicative inverse of the exponent (e) mod (p-1)*(q-1) proves the converse. that (e) is the multiplicative inverse of (d), and that (e) and (d) are both relatively prime to (p-1)*(q-1).

[18] *See* Stephen C. Pohlig and Martin E. Hellman, "An Improved Algorithm for Computing Logarithms over GF(p) and its Cryptographic Significance," IEEE TRANSACTIONS ON INFORMATION THEORY (Jan. 1978) (article submitted on June 17, 1976).

[19] The application for a patent on the Pohlig-Hellman system was pending during the period the RSA patent was being prosecuted, and the Patent Office declared an "interference" in which both sides were allowed to compete for a claim which would have covered the generic process of encryption and decryption using exponentiation and modular arithmetic. Stanford University and MIT abandoned the interference in the early 1980's, however, so neither side was awarded this generic claim.

[20] 35 U.S.C. section 271(c).

[21] *E.g., Joy Technologies, Inc. v. Flakt, Inc.*, 6 F.3d 770, 774 (Fed. Cir. 1993); *Met-Coil Systems Corp. v. Korners Unlimited, Inc.*, 803 F.2d 684, 687 (Fed. Cir. 1986). Inducing infringement also requires an intent element not present in a claim for contributory infringement. *Hewlett Packard Co. v. Bausch & Lomb, Inc.*, 909 F.2d 1464, 1468 (Fed. Cir. 1990).

[22] One cannot contribute to the infringement if the "direct" infringer is licensed, expressly or impliedly. *See, e.g., Universal Electronics, Inc. v. Zenith Electronics Corp.*, 846 F. Supp. 641 (N.D. Ill.) (no contributory infringement or inducement to infringe by manufacturer of "universal" remote control for television where the purchaser of the replacement remote has an implied license to practice the invention arising out of purchase of the original television-remote combination), *aff'd w/o op.*, 41 F.3d 1520 (Fed. Cir. 1994).

[23] *E.g., Hodosh v. Block Drug Co.*, 833 F.2d 1575, 1579 (Fed. Cir. 1987), *cert. denied*, 485 U.S. 1007 (1988).

[24] *See C.R. Bard, Inc. v. Advanced Cardiovascular Sys., Inc.*, 911 F.2d 670, 674 (Fed. Cir. 1990); *Universal Electronics*, 846 F. Supp. at 651.

[25] Other non-RSA public key variants, such as Diffie-Hellman and El-Gamal, all use exponentiation and modular reduction. *See, e.g.*, Schneier at 476-478; 513-516. While Pohlig-Hellman is the closest system to RSA, it is certainly true that the mathematical steps involved in RSA encryption and decryption are the same steps used in many other non-infringing cryptographic systems.

[26] The details of the SSL protocol may be found at http://home.netscape.com/eng/ssl3/ssl-toc.html.

[27] Netscape Navigator, for example, installs a file names "cert.db" with the other program files.

[28] Indeed, RSA Data itself operates a certificate authority business, and RSA certificate keys are supplied with the Netscape Navigator Browser.

[29] The basic patent statute, 35 U.S.C. section 101, provides, "[w]hoever invents or discovers any new and useful process, machine, manufacture, or composition of matter, or any new and useful improvement thereof, may obtain a patent therefor . . . ."

[30] *Gottschalk v. Benson*, 409 U.S. 63, 65 (1972).

[31] *Parker v. Flook*, 437 U.S. 584, 586 (1978).

[32] 450 U.S. 175 (1981).

[33] 573 F.2d 1237 (C.C.P.A. 1978).

[34] 618 F.2d 758 (C.C.P.A. 1980).

[35] 684 F.2d 902 (C.C.P.A. 1982).

[36] 958 F.2d 1053, 1058 (Fed. Cir. 1992).

[37] RSA Patent at Col. 14, Line 38 - Col. 15, Line 5.

[38] *Id.* at Col. 25, Lines 47-67.

[39] 33 F.3d 1526 (Fed. Cir. 1994).

[40] *Id.* at 1544.

[41] 33 F.3d 1354 (Fed. Cir. 1994).

[42] *Id.* at 1360-1.

[43] 22 F.3d 290 (Fed. Cir. 1994).

[44] *Id.* at 294 (emphasis in original). In *Warmerdam* (subsequent to *Schrader*), the input data was information about the physical objects the robot might encounter, and thus would also satisfy the "physical activity or object" test.

[45] *In re Schrader*, 22 F.3d at 294.

[46] *Id.* at 293-94.

[47] See RSA Patent claims 2, 4, 9, 14, 19, and 34.

[48] The United States Patent & Trademark Office will consider claims reciting generic computer hardware as process claims which themselves must be independently patentable. *See* United States Patent & Trademark Office, *Examination Guidelines for Computer-Related Inventions*, 61 Fed. Reg. 7478 (1996) (hereinafter, "Guidelines"). These Guidelines have been called "persuasive authority" by the Federal Circuit. *In re Trovato*, 60 F.3d 807 (Fed. Cir. 1995).

[49] These claims are stated in so-called "means-plus-function" terms, which is permissible under the last paragraph of 35 U.S.C. section 112. When claims are written in these terms, however, they are limited to covering "the corresponding structure . . . described in the specification and equivalents thereof." We have not analyzed whether the specific computational hardware disclosed in the specification of the patent differs in any material way from the processors used in modern computers, and it may well be that these claims are limited to 1977 era computational equipment. This article assumes, however, that a modern computer CPU would be considered at least the "equivalent" to the "register means" claimed in these dependent claims.

[50] This may be contrasted with the "preponderance-of-evidence" standard typical in other civil litigation.

[51] *In re Donaldson Co.*, 16 F.3d 1189, 1192 (Fed. Cir. 1994).

[52] *See generally Guidelines, supra* note 45.

[53] 35 U.S.C. section 112.

[54] *Amgen, Inc. v. Chugai Pharmaceutical Co.*, 927 F.2d 1200, 1209-10 (Fed. Cir.), *cert. denied*, 502 U.S. 856 (1991).

[55] *Amgen*, 927 F.2d at 1209; *Engel Industries, Inc. v. Lockformer Co.*, 946 F.2d 1528, 1531 (Fed. Cir. 1991).

[56] Ironically, it turns out that Dr. Rivest was wrong in his belief that these (and other) limitations on the selections of (p) and (q) made a significant difference in the security of the system. Subsequent research showed that, even with careful selection of (p) and (q), one could not avoid the sort of weakness in the modulus Dr. Rivest was hoping to avoid. *See* Alfred J. Menezes, Paul C. van Oorschot, Scott A. Vanstone, HANDBOOK OF APPLIED CRYPTOGRAPHY, section 3.2.4, p. 94 (CRC Press 1997). Users were told that a "careful" selection of primes allowed a given level of security to be achieved with a smaller modulus than would otherwise be required. In fact, this is an unsafe improvement, and the larger modulus is needed to achieve that level of security. Any impact of this fact on the validity of the patent is beyond the scope of this article.

[57] Ronald L. Rivest, Adi Shamir, Len Adelman, "A Method for Obtaining Digital Signatures and Public

Key Cryptosystems," MIT LABORATORY FOR COMPUTER SCIENCE TECHNICAL MEMORANDUM 82 (Revised August 1977) at 10.

[58] Ronald L. Rivest, "Remarks on a Proposed Cryptanalytic Attack on the M.I.T. Public-Key Cryptosystem," CRYPTOLOGIA (January 1978).

[59] *Id.* at 2.

[60] *Id.* at 4-5.

[61] *See Dana Corp. V. IPC Limited Partnership*, 860 F.2d 415, 418-419 (Fed. Cir. 1988), *cert. denied*, 490 U.S. 1067 (1989); *Advanced Semiconductor Materials America, Inc. v. Applied Materials, Inc.* 1994 W.L. 715634 (N.D. Cal. Dec. 16, 1994) (denying motion for summary judgment on best mode defense, ruling that the PTO Manual of Patent Examining Procedure, which specifically disallows incorporation by reference to "non patent publications" to comply with the best mode requirement, established the legal rule for incorporation by reference).

[62] *Graham v. John Deere Co.*, 383 U.S. 1, 14-15 (1966).

[63] See note 18 above.

[64] In 1870, a book by William S. Jevons described the relationship of one-way functions to cryptography *and* went on to discuss specifically the factorization problem used to create the "trap-door" in the RSA system. In July, 1996, one observer commented on the Jevons book in this way:

> In his book *The Principles of Science: A Treatise on Logic and Scientific Method,* written and published in the 1890's, William S. Jevons observed that there are many situations where the 'direct' operation is relatively easy, but the 'inverse' operation is significantly more difficult, One example mentioned briefly is that enciphering (encryption) is easy while deciphering (decryption) is not. In the same section of *Chapter 7: Introduction* titled 'Induction an Inverse Operation', much more attention is devoted to the principle that multiplication of integers is easy, but finding the (prime) factors of the product is much harder. *Thus, Jevons anticipated a key feature of the RSA Algorithm for public key cryptography, though he certainly did not invent the concept of public key cryptography.*
>
> Solomon W. Golomb, *On Factoring Jevons' Number*, CRYPTOLOGIA 243 (July 1996) (emphasis added).

This article is presented by CyberLaw (tm), an educational service on computer law for computer users. For more information on CyberLaw, contact Jonathan Rosenoer at cyberlaw@cyberlaw.com. Questions and comments may be posted at the CyberLaw Internet site (http://www.cyberlaw.com/). CyberLaw is a trademark of Jonathan Rosenoer. Copyright © 1997 Jonathan Rosenoer; All Rights Reserved.

FIG. 1

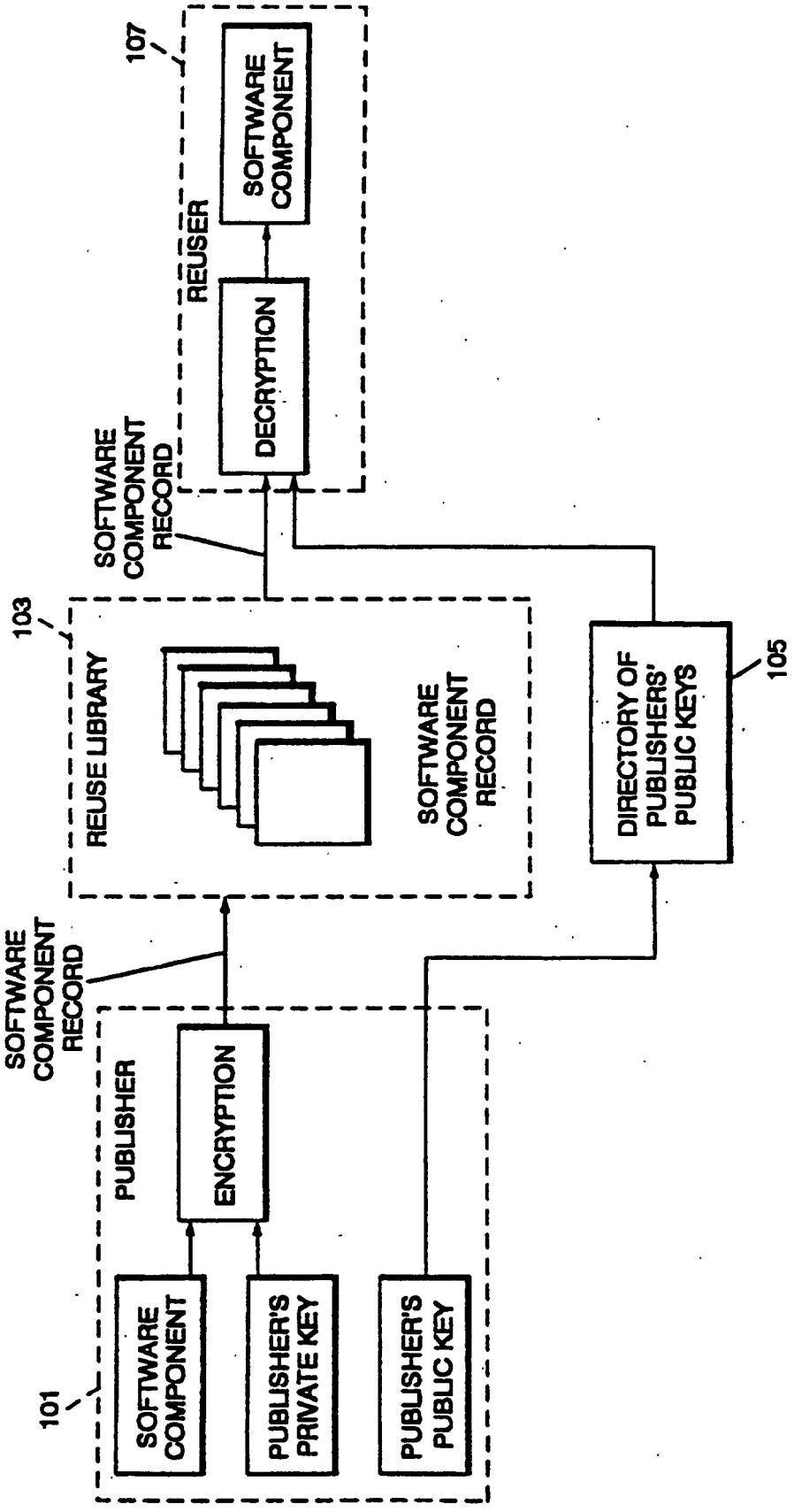

107
SOFTWARE COMPONENT
REUSER
DECRYPTION
SOFTWARE COMPONENT RECORD
103
REUSE LIBRARY
SOFTWARE COMPONENT RECORD
DIRECTORY OF PUBLISHERS' PUBLIC KEYS
105
SOFTWARE COMPONENT RECORD
PUBLISHER
ENCRYPTION
101
SOFTWARE COMPONENT
PUBLISHER'S PRIVATE KEY
PUBLISHER'S PUBLIC KEY

FIG. 2

FIG. 2A	FIG. 2B

FIG. 2A

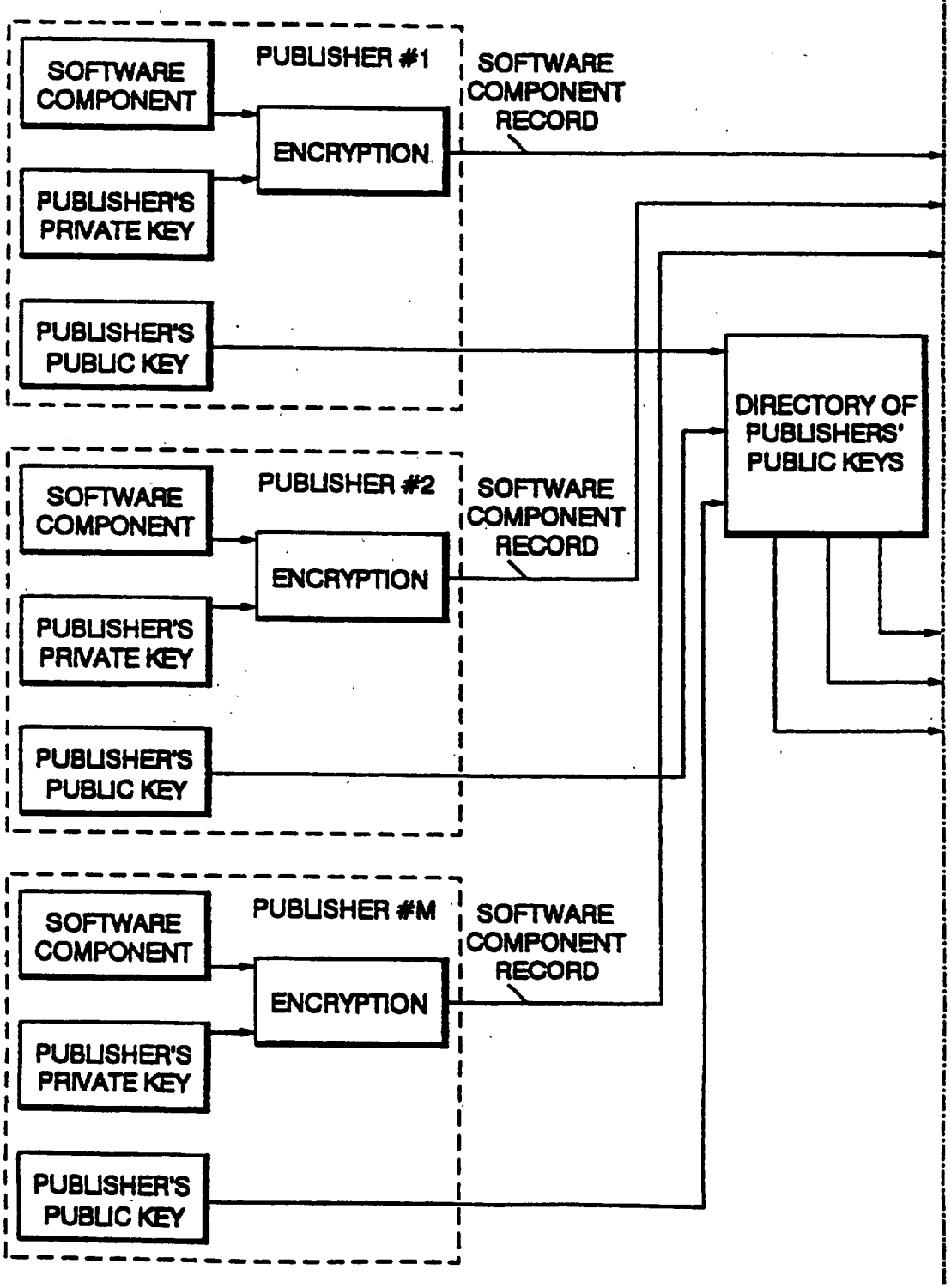
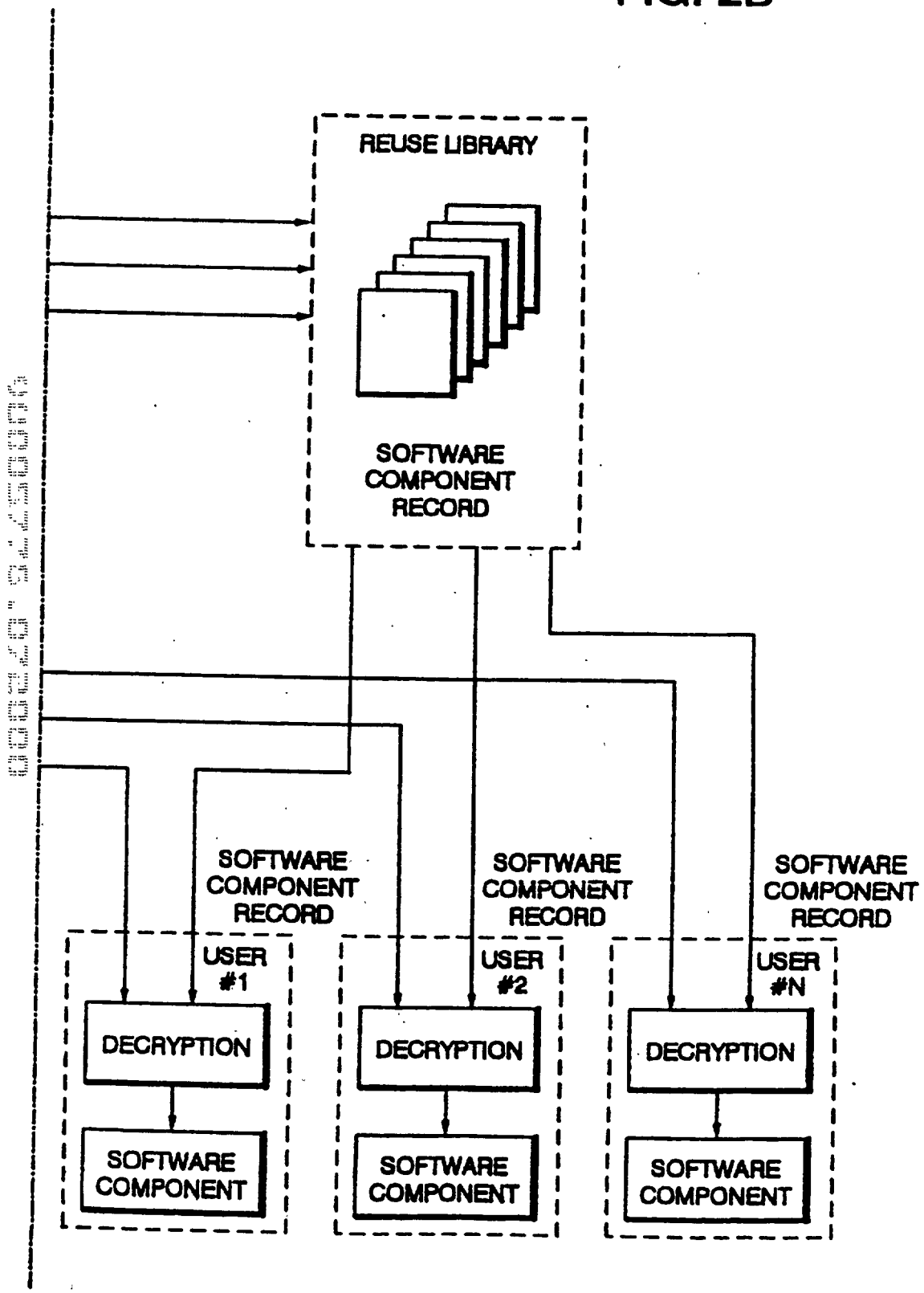


FIG. 2B


REUSE LIBRARY
SOFTWARE COMPONENT RECORD
SOFTWARE COMPONENT RECORD
SOFTWARE COMPONENT RECORD
SOFTWARE COMPONENT RECORD
USER #1
USER #2
USER #N
DECRYPTION
DECRYPTION
DECRYPTION
SOFTWARE COMPONENT
SOFTWARE COMPONENT
SOFTWARE COMPONENT

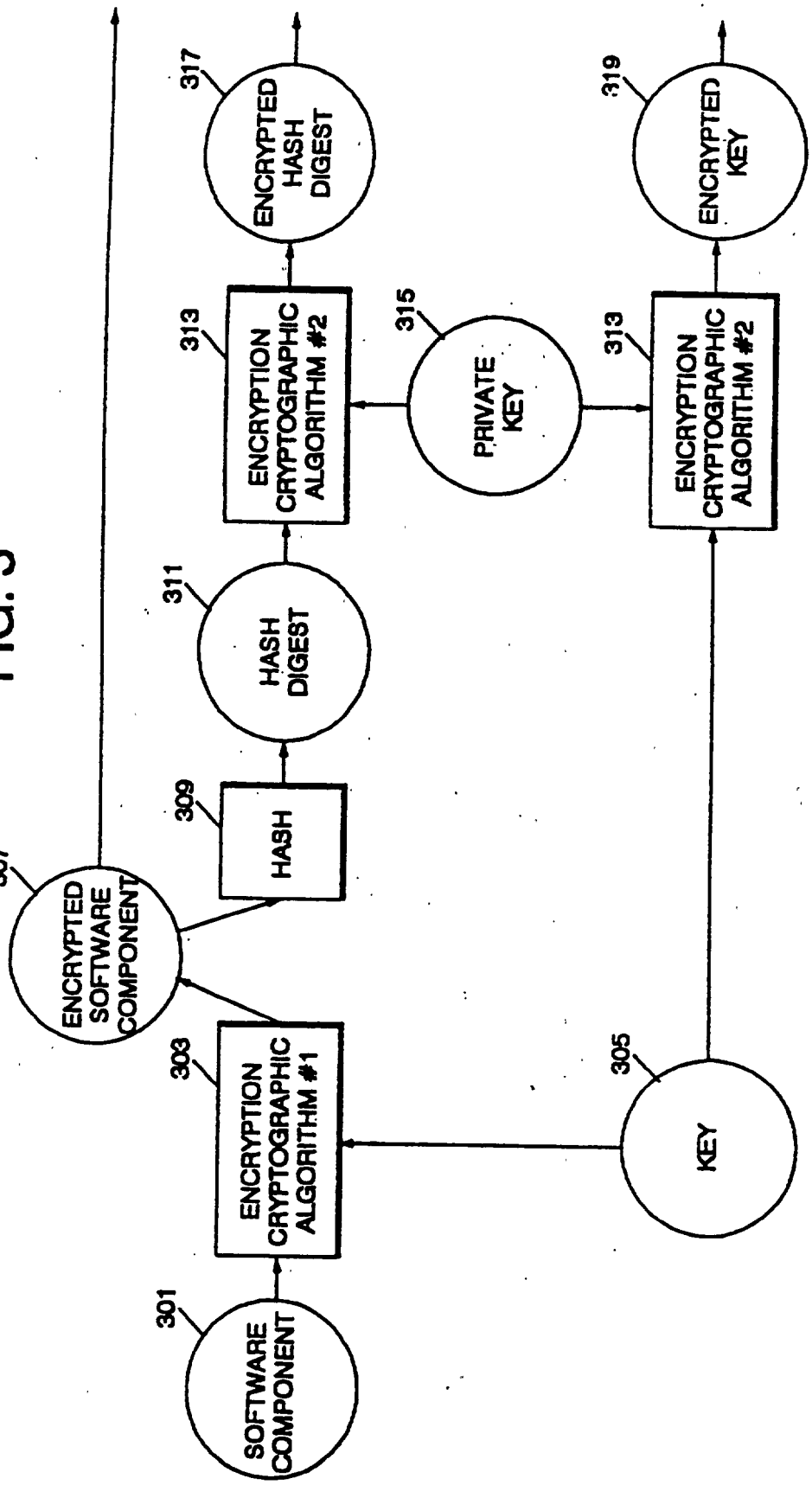

SOFTWARE COMPONENT
301
ENCRYPTION CRYPTOGRAPHIC ALGORITHM #1
303
KEY
305
ENCRYPTED SOFTWARE COMPONENT
307
HASH
309
HASH DIGEST
311
ENCRYPTION CRYPTOGRAPHIC ALGORITHM #2
313
PRIVATE KEY
315
ENCRYPTED HASH DIGEST
317
ENCRYPTION CRYPTOGRAPHIC ALGORITHM #2
313
ENCRYPTED KEY
319

FIG. 3

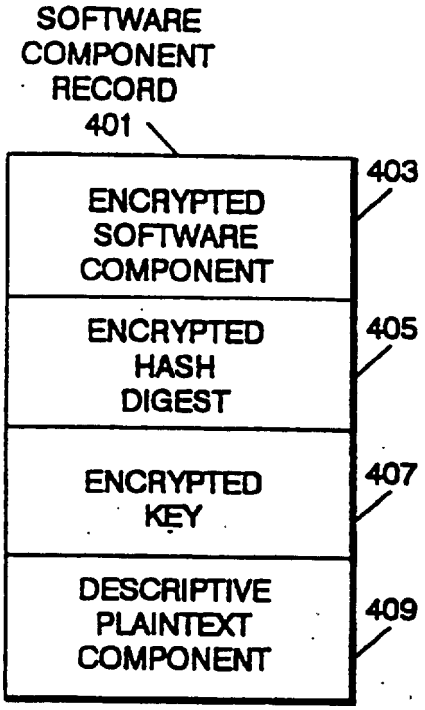

SOFTWARE COMPONENT RECORD
401
ENCRYPTED SOFTWARE COMPONENT
403
ENCRYPTED HASH DIGEST
405
ENCRYPTED KEY
407
DESCRIPTIVE PLAINTEXT COMPONENT
409

FIG. 4

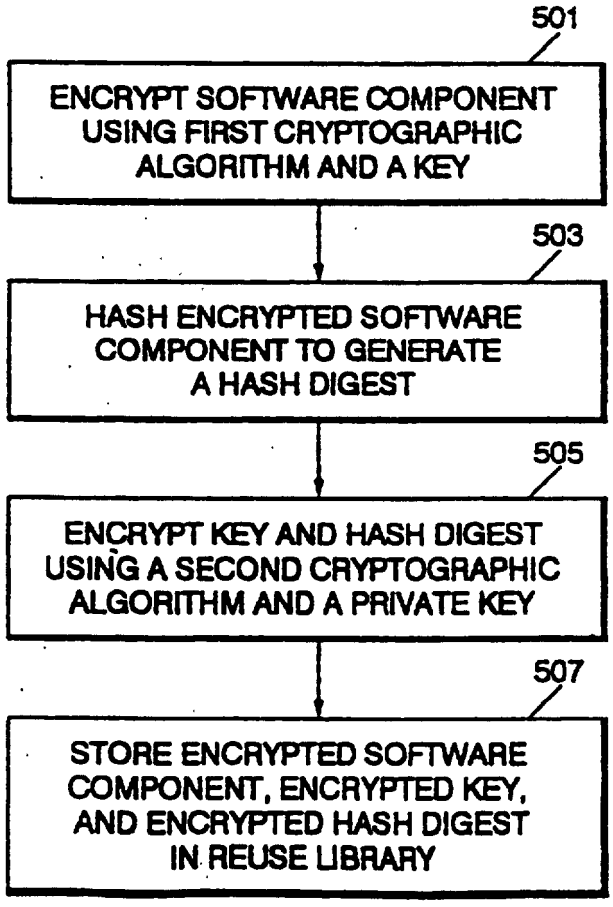

501
ENCRYPT SOFTWARE COMPONENT USING FIRST CRYPTOGRAPHIC ALGORITHM AND A KEY
503
HASH ENCRYPTED SOFTWARE COMPONENT TO GENERATE A HASH DIGEST
505
ENCRYPT KEY AND HASH DIGEST USING A SECOND CRYPTOGRAPHIC ALGORITHM AND A PRIVATE KEY
507
STORE ENCRYPTED SOFTWARE COMPONENT, ENCRYPTED KEY, AND ENCRYPTED HASH DIGEST IN REUSE LIBRARY

FIG. 5

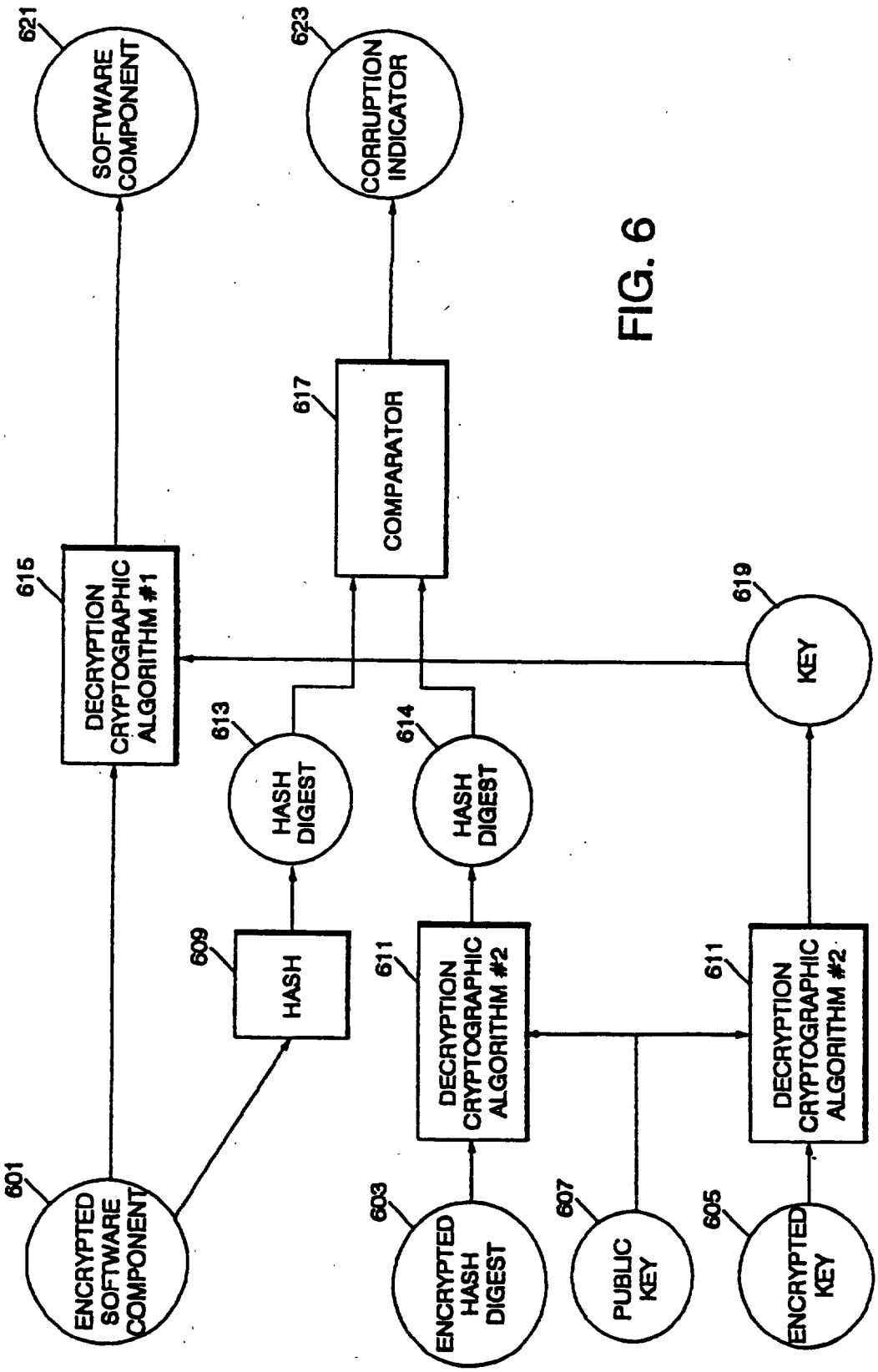

621
SOFTWARE COMPONENT
623
CORRUPTION INDICATOR
617
COMPARATOR
615
DECRYPTION CRYPTOGRAPHIC ALGORITHM #1
619
KEY
613
HASH DIGEST
614
HASH DIGEST
609
HASH
611
DECRYPTION CRYPTOGRAPHIC ALGORITHM #2
611
DECRYPTION CRYPTOGRAPHIC ALGORITHM #2
601
ENCRYPTED SOFTWARE COMPONENT
603
ENCRYPTED HASH DIGEST
607
PUBLIC KEY
605
ENCRYPTED KEY

FIG. 6

FIG. 7

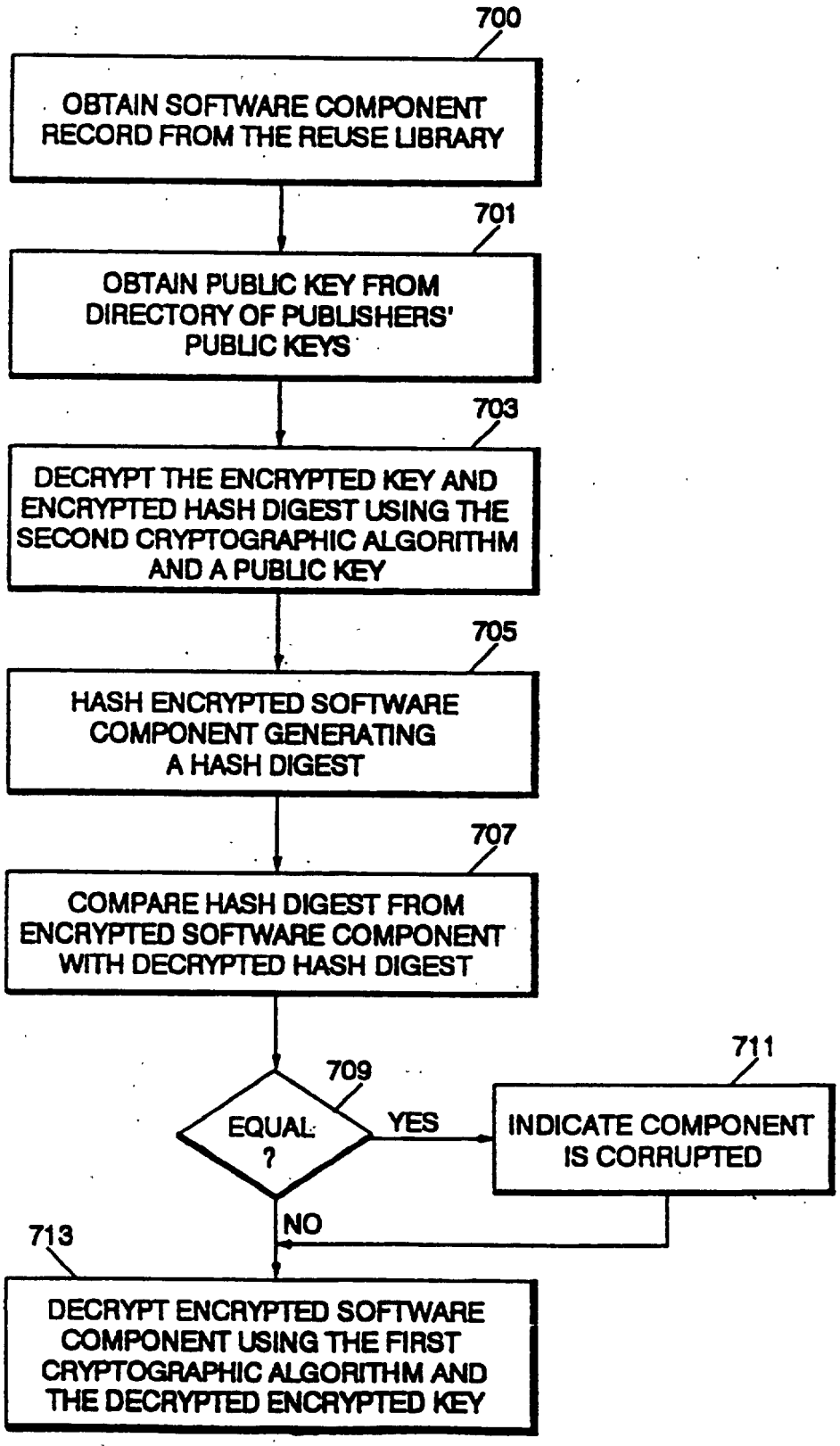

700
OBTAIN SOFTWARE COMPONENT RECORD FROM THE REUSE LIBRARY
701
OBTAIN PUBLIC KEY FROM DIRECTORY OF PUBLISHERS' PUBLIC KEYS
703
DECRYPT THE ENCRYPTED KEY AND ENCRYPTED HASH DIGEST USING THE SECOND CRYPTOGRAPHIC ALGORITHM AND A PUBLIC KEY
705
HASH ENCRYPTED SOFTWARE COMPONENT GENERATING A HASH DIGEST
707
COMPARE HASH DIGEST FROM ENCRYPTED SOFTWARE COMPONENT WITH DECRYPTED HASH DIGEST
709
EQUAL ?
YES
711
INDICATE COMPONENT IS CORRUPTED
NO
713
DECRYPT ENCRYPTED SOFTWARE COMPONENT USING THE FIRST CRYPTOGRAPHIC ALGORITHM AND THE DECRYPTED ENCRYPTED KEY

FIG. 8

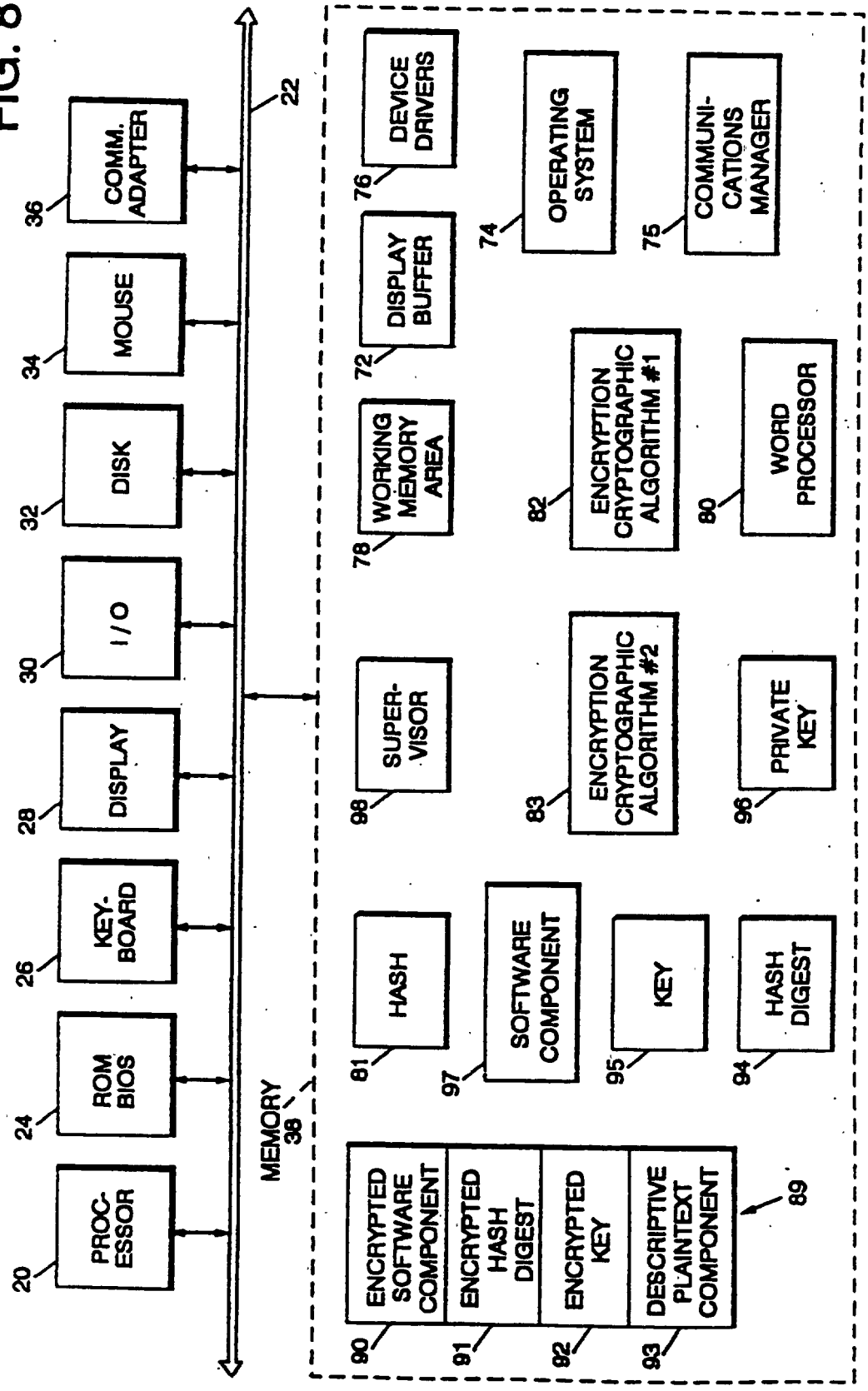

PROC-ESSOR
20
ROM BIOS
24
KEY-BOARD
26
DISPLAY
28
I/O
30
DISK
32
MOUSE
34
COMM. ADAPTER
36
22
MEMORY
38
DEVICE DRIVERS
76
DISPLAY BUFFER
72
WORKING MEMORY AREA
78
SUPER-VISOR
98
HASH
81
SOFTWARE COMPONENT
97
KEY
95
HASH DIGEST
94
PRIVATE KEY
96
ENCRYPTION CRYPTOGRAPHIC ALGORITHM #2
83
ENCRYPTION CRYPTOGRAPHIC ALGORITHM #1
82
WORD PROCESSOR
80
OPERATING SYSTEM
74
COMMUNI-CATIONS MANAGER
75
ENCRYPTED SOFTWARE COMPONENT
90
ENCRYPTED HASH DIGEST
91
ENCRYPTED KEY
92
DESCRIPTIVE PLAINTEXT COMPONENT
93
89

FIG. 9

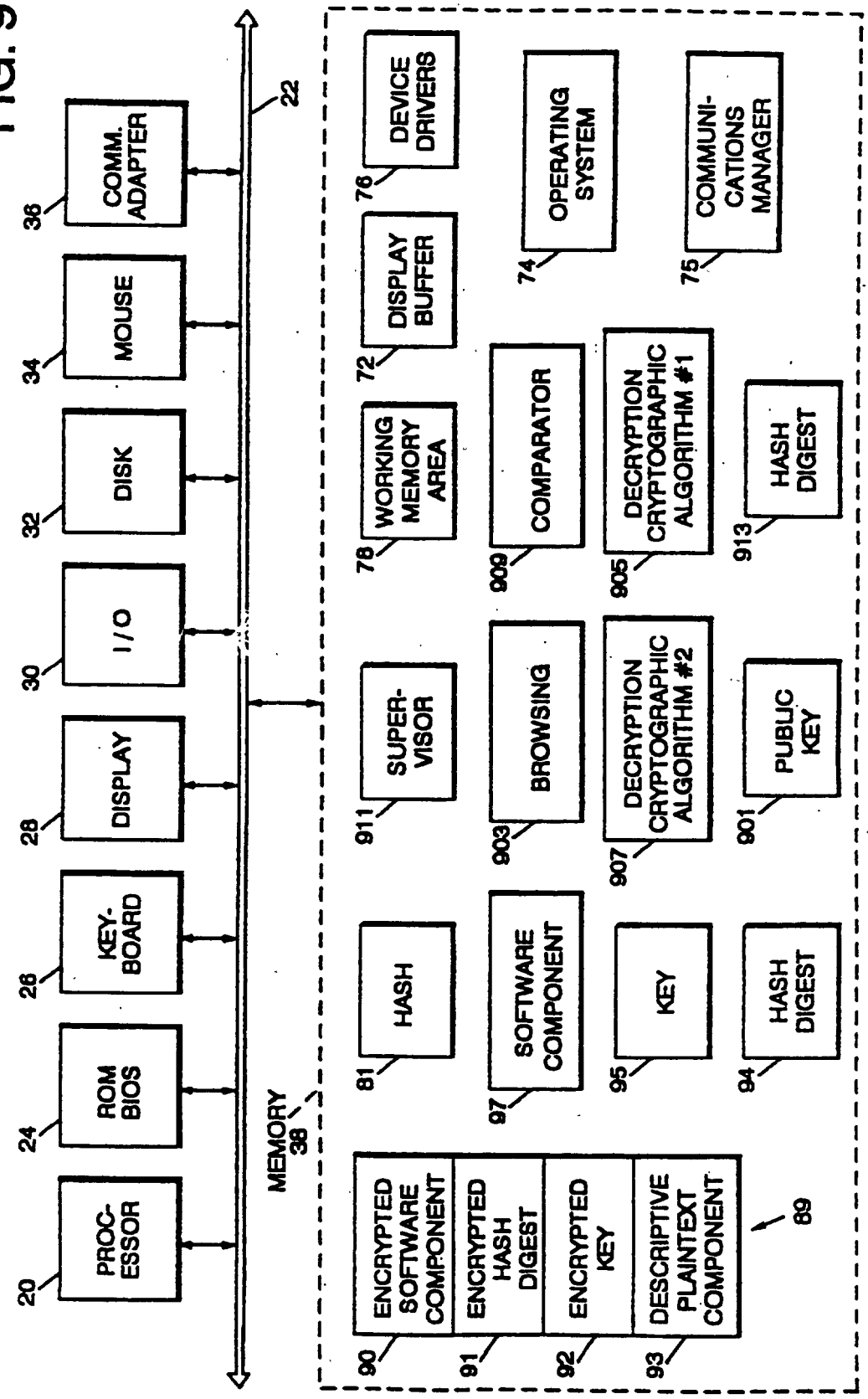
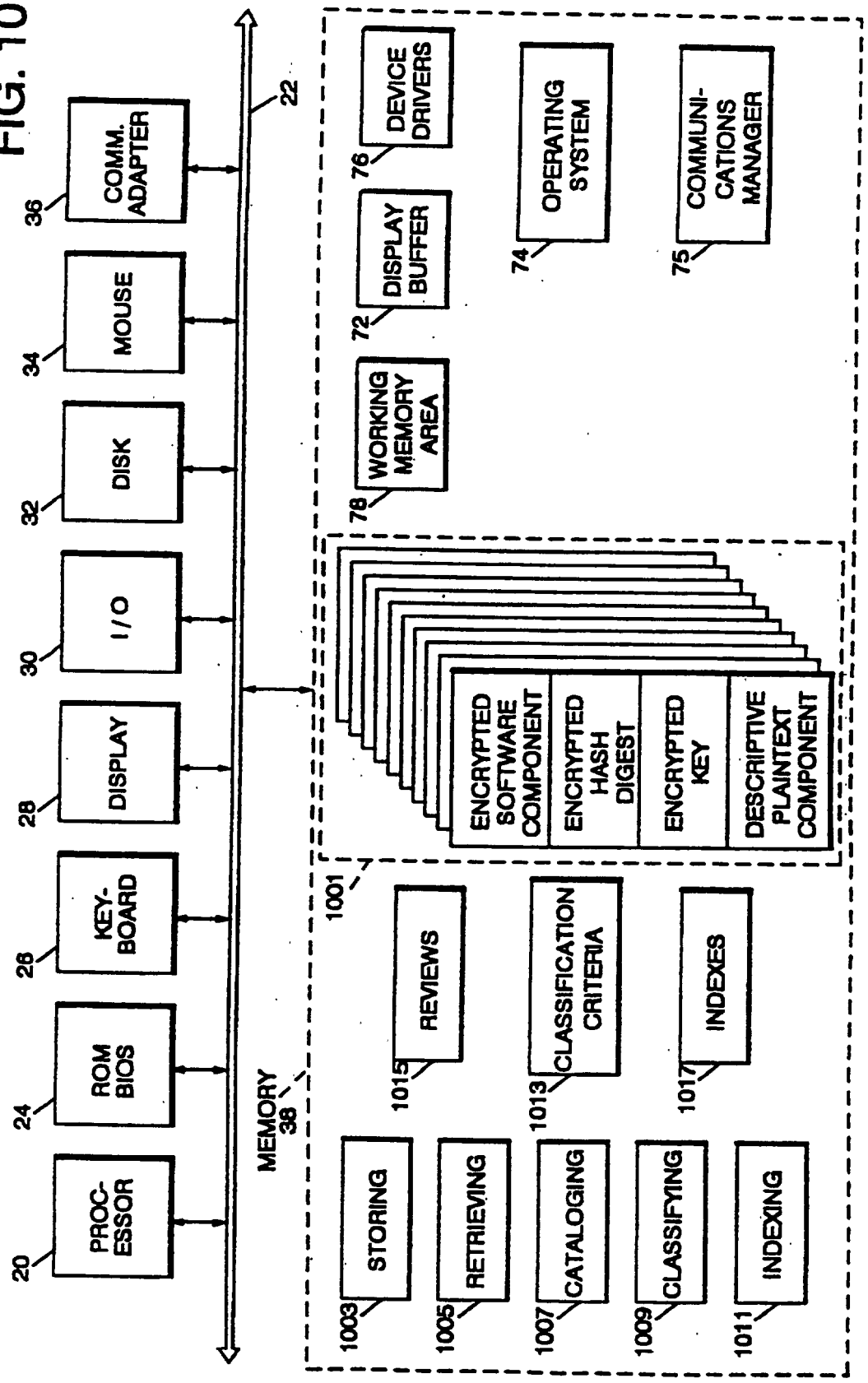

PROC-ESSOR
20
ROM BIOS
24
KEY-BOARD
26
DISPLAY
28
I/O
30
DISK
32
MOUSE
34
COMM. ADAPTER
36
22
MEMORY 38
DEVICE DRIVERS
76
DISPLAY BUFFER
72
WORKING MEMORY AREA
78
SUPER-VISOR
911
BROWSING
903
SOFTWARE COMPONENT
97
HASH
81
KEY
95
HASH DIGEST
94
PUBLIC KEY
901
DECRYPTION CRYPTOGRAPHIC ALGORITHM #2
907
COMPARATOR
909
DECRYPTION CRYPTOGRAPHIC ALGORITHM #1
905
HASH DIGEST
913
OPERATING SYSTEM
74
COMMUNI-CATIONS MANAGER
75
ENCRYPTED SOFTWARE COMPONENT
90
ENCRYPTED HASH DIGEST
91
ENCRYPTED KEY
92
DESCRIPTIVE PLAINTEXT COMPONENT
93
89

FIG. 10


PROC-ESSOR
20
ROM BIOS
24
KEY-BOARD
26
DISPLAY
28
I/O
30
DISK
32
MOUSE
34
COMM. ADAPTER
36
22
MEMORY
38
DEVICE DRIVERS
76
DISPLAY BUFFER
72
WORKING MEMORY AREA
78
OPERATING SYSTEM
74
COMMUNI-CATIONS MANAGER
75
1001
ENCRYPTED SOFTWARE COMPONENT
ENCRYPTED HASH DIGEST
ENCRYPTED KEY
DESCRIPTIVE PLAINTEXT COMPONENT
REVIEWS
1015
CLASSIFICATION CRITERIA
1013
INDEXES
1017
STORING
1003
RETRIEVING
1005
CATALOGING
1007
CLASSIFYING
1009
INDEXING
1011

HYBRID ENCRYPTION METHOD AND SYSTEM FOR PROTECTING REUSABLE SOFTWARE COMPONENTS

BACKGROUND OF THE INVENTION

1. Field of the Invention

The present invention relates to a system and method for protecting reusable software components. In particular the present invention encrypts software components in a reusable software component library to protect against unauthorized modification and to assure authenticity of software components when the software components are decrypted by a reuser.

2. Description of the Prior Art

The ability to produce ever larger software systems while improving theft quality and reducing theft development time crucially depends upon a capability to "reuse" previously developed software components in new systems. There is an emerging electronic marketplace that will enable potential reusers to browse libraries of software components, select suitable ones, and obtain them for reuse. Rudiments of such a marketplace already exist in operational software reuse libraries. Such libraries, though, are vulnerable to the unauthorized modification of existing code or to unscrupulous parties who might misrepresent the origin of code which they place into the library.

A reuse library must provide protection against the unauthorized modification of software components in order for the software reuse marketplace to emerge. Without such protection publishers would be reluctant to place their software in a reuse library and reusers would be reluctant to use software components from the reuse library. Without such protection software components are subject to modification for purposes of malice, sabotage, espionage or others. Modifications by an innocent third party can also cause problems due to incompetence, carelessness, a lack of discipline or misunderstanding. The ability of a third party to modify a software component without detection cannot be tolerated in the reuse marketplace.

The authentication problem arises where an unscrupulous party seeks to pass off (or palm off) their software components as that of another publisher thereby preying on the reputation and goodwill of other publishers. This is of particular importance in the reuse marketplace because reusers can often only rely on the reputations and software development processes used by software publishers.

Many agencies today are actively involved in developing and evolving their software development processes. Independent organizations such as the Software Engineering Institute (SEI) evaluate these processes and rate them according to an established set of criteria. Reusers can rely on these evaluations in making their reuse selections.

There are also legal considerations to be considered such as who is representing that they created the software. Under current copyright law the innocent infringer loses against the true owner of the copyrighted work. The reuser needs some assurance that publisher has the right to permit the reuser to make use of the software component. Without such assurances the reuser risks any gains by reuse in a subsequent legal battle.

There is a critical need to provide a reuser with an assurance that the identity of the producer of software

component being purchased is the software producer and not some other third party. There is also a critical need to provide the reuser with an assurance that a software component has not been corrupted or modified.

SUMMARY OF THE INVENTION

The present invention is directed to a method and apparatus that satisfies these needs. It is an object of the invention to provide an integrity mechanism that provides an indication that a software component has been modified.

It is an object of the invention to quickly provide an integrity mechanism that provides an indication that a software component has been modified. It is a still a further object to prevent third parties from modifying software components in a reuse library. It is an object of the invention to provide an authentication mechanism that provides reusers with assurance that the software component is the authentic product of its stated publisher. It is yet another object of the invention to prevent third parties from passing off their software components as that of another. It is still another object of the invention to provided for the authenticity of a software component and an indication of whether the software component has been modified.

Accordingly, the present invention provides a system and method for providing a reuser with an indication of whether or not a software component from a reuse library is authentic and whether or not the software component has been modified. The present invention comprises a method for reusing software components that maintains the integrity and authenticity of the software components. The method comprises generating an software component record by encrypting a plaintext representation of a software component into a encrypted software component with a first cryptographic algorithm using first key; hashing the encrypted software component to generate a first hash digest; encrypting the first hash digest and the first key using a second cryptographic algorithm with a second key, wherein said second cryptographic algorithm is of a public key type and said second key is the private key associated with at least one public key, said software component record consisting of the encrypted software component, the encrypted hash digest, and the encrypted first key. The software component record is then stored in a reuse library. The software component can then be retrieved from the reuse library. The plaintext representation of the software component is then generated by obtaining a public key associated with the second key from a public key directory; decrypting the encrypted hash digest and the encrypted first key into the decrypted first key and the decrypted first hash digest using the public key and the second cryptographic algorithm; hashing the encrypted software component to generate a second hash digest; comparing the second hash digest with the decrypted first hash digest, and if not identical indicating that the software component is corrupted, if identical indicating that the software is not corrupted; decrypting the encrypted software component into the plaintext representation using the decrypted first key and the first encryption algorithm.

The present invention comprises a network of computer systems comprising a reuse library, a directory, at least one publisher's workstation and at least one reuser's workstation. The reuse library having a plurality

of encrypted software components each software component record having an encrypted software component, an encrypted hash digest, and an encrypted first key; the reuse library also having a storage means for storing encrypted software components and a retrieval means for retrieving encrypted software components. The directory containing a list of publishers and an associated list of public keys. A publisher's workstation coupled to the reuse library, having an first encrypting means for encrypting a plaintext representation of a software component into a encrypted software component with a first cryptographic algorithm using first key; an hashing means for hashing the encrypted software component to generate a first hash digest; a second encrypting means for encrypting the first hash digest and the first key using a second cryptographic algorithm with a second key, wherein said second cryptographic algorithm is of a public key algorithm type and said second key is the publisher's private key associated with a publisher's public key, said software component record consisting of the encrypted software component, the encrypted hash digest, and the encrypted first key; a communications means for sending the software component record to the reuse library for storage by the storage means. A reuser workstation coupled to the reuse library, said reuser workstation having a requesting means for sending a request to the reuse library for a desired encrypted software component, wherein said request causes the retrieval means of the reuse library to retrieve the desired software component and send it to the requesting workstation; a means for obtaining the public key from the directory, said public key associated with the second key of the desired encrypted software component; a first decrypting means for decrypting the encrypted hash digest and the encrypted first key into the decrypted first key and the decrypted first hash digest using the public key and the second cryptographic algorithm; an hashing means for hashing the encrypted software component to generate a second hash digest; a comparing means for comparing the second hash digest with the decrypted first hash digest, and if not identical indicating that the software component is corrupted, if identical indicating that the software is not corrupted; a second decrypting means for decrypting the encrypted software component into the plaintext representation using the decrypted first key and the first encryption algorithm.

BRIEF DESCRIPTION OF THE DRAWINGS

The foregoing and other objects, aspects and advantages of the invention will Inc better understood from the following detailed description with reference to the drawings, in which:

FIG. 1 shows a functional overview of the present invention.

FIG. 2 shows a functional overview of the present invention depicting multiple publishers and multiple reusers.

FIG. 3 depicts generating various components of a software component record.

FIG. 4 depicts a representation of a software component record.

FIG. 5 depicts the steps required to generate various components of a software component record.

FIG. 6 depicts generation of the plaintext representation of a software component from a software component record.

FIG. 7 depicts the steps required to generate a plaintext representation of a software component from a software component record.

FIG. 8 depicts one embodiment of a computer system for generating a software component record.

FIG. 9 depicts one embodiment of a computer system for generating a software component from a software component record.

FIG. 10 depicts one embodiment of a computer system for the reuse library.

DETAILED DESCRIPTION OF THE INVENTION

I. DEFINITIONS
II. OVERVIEW
III. CRYPTOGRAPHIC ALGORITHMS & FUNCTIONS
A. OVERVIEW
B. PUBLIC KEY and CONVENTIONAL CRYPTOGRAPHIC SYSTEMS
C. EXAMPLE CRYPTOGRAPHIC ALGORITHMS
1. Conventional Algorithms
2. Public Key Algorithms
D. HASHING
IV. PUBLISHING
A. DESCRIPTION
B. EMBODIMENTS
V. REUSING
A. DESCRIPTION
B. BROWSING MEANS
C. EMBODIMENTS
VI. REUSE LIBRARY
A. DESCRIPTION
B. EMBODIMENTS
VII. DIRECTORY
VIII. ADVANTAGES AND CLOSING

I. DEFINITIONS

A. "SOFTWARE COMPONENT" is a set of statements or instructions to be used directly or indirectly in a computer in order to bring about a certain result. Thus, a software component can consist of a complete software application, a set of related applications, a module, a single procedure or program, a set of procedures or programs, a software package or a set of software packages. It is preferable for reuse software components to be provided in source code in human readable format.

II. OVERVIEW

FIG. 1 shows the major elements of the present invention a publisher 101, a reuse library 103, a directory 105 and a reuser 107. The reuse library 103 can contain many software components. The software components are created by software providers. These software providers, called publishers herein, are responsible for the creation of software components. The act of placing the software component into the reuse library is referred herein as publishing the software component. Once placed in the reuse library by a publisher, the software component becomes available for reuse by other entities. These other entities may consist of individuals, corporations, associations, government branches agencies or departments. The reuser 107 must decide whether a software component placed in the reuse library is suitable for an application or software requirement that they may have. The reuser can browse soft-

ware components in the reuse library while determining whether any software components suit the reuser's particular requirements. By making use of software components in the reuse library the reuser can reduce their software development costs. The act or process whereby a reuser selects a software component for use in an application or software effort is herein referred to as reusing. Reusing includes browsing of software components in the reuse library.

Although FIG. 1 shows only one publisher 101 and one reuser 107 the present invention contemplates many publishers and many reusers. In fact publishers may also be reusers and reusers may also be publishers. Multiple publishers and multiple reusers are shown in FIG. 2. The reuse library 103 may also consists of multiple libraries where each library is orientated towards particular classes of reusers. For instance, based on language type (e.g., C or Ada) or based on the type of application (e.g., real time systems or embedded systems) or by application or function (e.g., accounting, navigation, air traffic control, word processing, etc.). Thus, a multitude of reuse libraries are contemplated by the present invention. The reuse library may also perform certain classifying or cataloging functions so that a software component is indexed properly or more easily located by a potential user.

When the publisher 101 decides that a particular software component is ready for the reuse library 103 several steps must be taken so that an eventual reuser is assured that it is the particular publisher's software component and not an impostor's and that the software component has not been modified. The software component as developed by the publisher typically consists of a plaintext representation. This is typically an ASCII or EBCDIC encoded representation. An operator can thus view the software component on a display screen or print the software component on a printer. The plaintext representation is the unencrypted format. Before transmitting or sending the software component to the reuse library the software component is encrypted using the hybrid encryption technique of the present invention. Two cryptographic algorithms are used to encrypt the software component a conventional key algorithm and a public key algorithm. The encrypted software component is then sent to the reuse library for storage and eventual retrieval by a reuser. The encrypting or enciphering method assures that any reuser of the software component is provided with notice that the software component has been modified or that the software component is not authentic (i.e., that the publisher associated with the software component in the reuse library is not in fact the actual publisher). The publisher encrypts the software component using the hybrid technique using the publisher's private cryptographic key. Only the publisher knows the private key. This key must be safeguarded by the publisher if they are to assure the integrity of their software components. In order for the encrypted software component to be decrypted the reuser must use the publisher's public key. The public key is associated with the publishers private key, but cannot be used to derive the private key.

In order for the reuser to make of use of an encrypted software component the reuser must decrypt the encrypted software component from its encrypted, into its plaintext representation. The present invention requires that the reuser have the publishers public key in order to obtain the plaintext representation of the software component. The reuser obtains the publishers public key from the directory of publisher's public keys. The public key may be made available in a separate notebook, a traditional book, a separate file server, or as part of the reuse library. The directory 105 may also be provided on a trusted platform connected to the reuser by a trusted path to assure that only authorized reusers are provided with access to the reuse library. With the public key the reuser is able to obtain the plaintext representation of the software component.

III. CRYPTOGRAPHIC ALGORITHMS & FUNCTIONS

A. OVERVIEW

Cryptography is the transformation of intelligible information into apparently unintelligible form in order to conceal the information from unauthorized parties. Cryptography is a known practical method to protect information transmitted electronically through communications network and as will be shown with the present invention can be an economical way to protect stored data. The cryptographic transformation of data is defined by a cryptographic algorithm or procedure under the control of a value called a cryptographic key. See text "Cryptography and Data Security," by Denning, Addison-Wesley Publishing Company (1982).

Cryptographic methods can be used to protect not only the confidentiality of data, but the integrity of data as well. Data confidentiality is the protection of information from unauthorized disclosure. Data integrity is the protection of information from unauthorized modification.

There are two basic elements associated with any cryptographic system. These elements are a set of unchanging rules or steps called a cryptographic algorithm and a set of variable cryptographic keys. The algorithm is composed of encrypting and decrypting procedures which usually are identical or simply consist of the same steps performed in reverse order, but which can be dissimilar. The keys selected by the user consist of a sequence of numbers or characters. An encryption key (Ke) is used encrypt plaintext X into ciphertext Y as shown below

$$E_{Ke}(X)=Y$$

and a decryption key (Kd) is used to decryption ciphertext Y into plaintext X as shown below.

$$D_{Kd}[E_{Ke}(X)]=D_{Kd}[Y]=X$$

B. PUBLIC KEY and CONVENTIONAL CRYPTOGRAPHIC SYSTEMS

There are two basic types of Cryptographic algorithms: conventional and public key (also referred to as symmetric and asymmetric). With a conventional algorithm the encryption and decryption keys may either be easily computed from each other or the keys may be identical (Ke=Kd=K). In a public key algorithm, one key (usually the encryption key) is made public and a different key (usually the decryption key) is kept private. As will be discussed in detail below the present invention utilizes the private key to encrypt and the public key to decrypt. With a public key system it must not be possible to deduce the private key from the public key. When an algorithm is made public, for example with a published encryption standard, cryptographic

security completely depends on protecting these cryptographic keys.

To keep information secret, and to achieve privacy, a reversible algorithm must be used. This allows for reversing the encryption process to recover the software component or data item. However, encryption alone is insufficient to assure that information is not altered during storage. This is most evident when encryption with a public key algorithm is used. With a public key system as used with the present invention, any one can decrypt using the public decryption key, unlike public key systems where the public key is the encryption key and the private key is the decryption key and any system user or node can masquerade as any other system user or node.

In contrast to the conventional cryptographic algorithms a public key method uses two different keys to encrypt and decrypt a message. Successful methods are designed so that neither key may be inferred from the other. When used for authentication, the sender encrypts messages using the encryption key which is held in secrecy. The decryption key is made publicly known. Any receiver can decrypt the message using the publicly known key and be confident that the data is not forged or altered because only the presumed sender knows the corresponding encryption key.

In general it is preferable for performance reasons to use conventional algorithms such as DES for bulk data encryption rather than to use a public key algorithm.

The Digital Signature Algorithm (DSA) "hashes" the item to be authenticated so that a smaller "hash digest" is produced. The original item is transmitted in plaintext along with an encrypted version of the hash digest. The receiver authenticates by hashing the received plaintext to regenerate the hash digest, decrypts the transmitted hash digest and compares the two digests for equality.

This method has some weaknesses when applied to software components. First, because the component is available in plaintext, it is tempting for potential reusers to forget about authenticating the component and to simply use the plaintext as is. One could remedy this defect by encrypting the entire component. The problem with this approach is that public key cryptographic schemes are very slow in operation (as compared to private key schemes); they may be impractically slow when applied to objects as large as software components with their related documentation.

The solution to this problem is to use a hybrid encryption scheme of the present invention. With this approach, the software component is encrypted using a private-key method, like DES. The DES key could be generated in an arbitrary fashion for each usage so that, in practical terms, every usage is unique. The key that was used for a software component is included with the encrypted software component. The encrypted software component is hashed to generate a digest. The digest and the DES key are then encrypted using the public key method.

Reusers who wish to look at the software component decrypt the hash digest and the DES key by applying the publicly known decryption key of the software component's publisher. The reuser regenerates the hash digest from the encrypted software component and compare it with the just-decrypted hash digest. The reuser can use the just-decrypted DES key to quickly decrypt the encrypted software component.

C. EXAMPLE CRYPTOGRAPHIC ALGORITHMS

1. Conventional Algorithms

Data Encryption Standard (DES)

The *Data Encryption Standard* (*DES*) is described in Federal Information Processing Standard Publication (FIPS PUB) 46 and available from the National Technical Information Service, 5285 Port Royal Road, Springfield, Va. 22161. DES hardware is available from Technical Communications Corp.

SkipJack

Skipjack is a symmetric key algorithm viewed as a possible replacement to DES. Capstone is a data security chip that utilizes the skipjack algorithm, secure hash algorithm and the key exchange algorithm. {need a reference}

2. Public Key Algorithms

Public key algorithms are described in a paper by W. Diffie and M. E. Hellman entitled "Privacy and Authentication: An Introduction to Cryptography," Proceedings of the IEEE, Vol. 67, No. 3, March 1979, pp 397–427 hereby incorporated by reference. Examples of public key cryptographic systems are provided in: U.S. Pat. No. 4,218,582 to Hellman et at. "Public Key Cryptographic Apparatus and Method" and U.S. Pat. No. 4,200,770 to Hellman et al. "Cryptographic Apparatus and Method" hereby incorporated by reference.

Digital Signature Standard (DSS)

The National Institute of Standards and Technology has proposed a method for generating control signatures based on a 1985 paper by T. Eigamal entitled "A Public Key Cryptosystem and a Signature Scheme Based on Discrete Logarithms," IEEE Transactions on Information Theory, 31, 1985, pp. 469–472, hereby incorporated by reference. The DSS proposes use of the Digital Signature Algorithm (DSA) to guarantee authenticity and integrity of data transactions.

Rivest-Shamir-Aldeman (RSA)

The RSA public key algorithm is described in U.S. Pat. No. 4,405,829 to Rivest et. at. "Cryptographic Communications System and Method" herein incorporated by reference discloses the RSA public key algorithm.

D. HASHING

A one-way function is a function which is easy to compute in the forward direction, but hard to compute in the reverse direction. That is, if $Y = f(X)$ is a one-way function then given any X it is easy to compute the corresponding Y, taking typically a fraction of a second on a small computer. But given any Y it is extremely difficult to find the corresponding X, ideally taking millions of years on the most powerful computer imaginable. A one-way function can be expansionary (i.e., Y is longer than X), compressive, or neither, depending on the relative sizes of the ciphertext (Y) and key (X). For purposes of this invention, we are primarily concerned with one-way compressive functions, where X is much longer than Y. Typical values herein will be a 100,000 bit length for X and a 100 bit length for Y. A method for generating such an extremely compressive one-way function are well known in the art. Compressive functions are also called "hash functions" and a one-way

compressive function is therefore called a one-way hash function.

A method for deriving a one-way function from a conventional cryptographic system is described in section V of Diffie and Hellman's paper, "New Directions in Cryptography", IEEE Transactions on Cryptography, vol. IT-22, November 1976 (see, especially FIG. 3 therein). If X is the plaintext representation of a software component and $E_{Ke}(X)$ represents encrypted version of X using an encryption algorithm E with a cryptographic key Ke. Using a hashing function H, the hash digest H_d is defined as follows:

$$H_d = H(E_{Ke}(X))$$

Computing H_d from X merely involves an encryption $E_{Ke}(X)$ and computing the hashing function (H) given the encrypted software component $E_{Ke}(X)$ and is therefore a simple computation. But computing $E_{Ke}(X)$ or X from H_d involves cryptoanalysis because $X=H^{-1}(H_d)$ and is therefore difficult to compute.

One way hashing functions are well known in the art. Hashing functions suitable for use with the present invention are described in U.S. Pat. No. 4,924,515 titled "Secure Management of Keys Using Extended Control Vectors" to Matyas et. al and U.S. Pat. No. 4,908,861 titled "Data Authentication Using Modification Detection Codes" to Brachtl et. al which are hereby incorporated by reference.

IV. PUBLISHING

A. DESCRIPTION

When a publisher decides that a software component is ready for publishing several steps must be taken to produce the software component record, as depicted in FIG. 4. Referring now to FIG. 3, which depicts data objects as circles and functions as rectangles, we start with a software component 301 in a plaintext representation. This plaintext representation is typically source code and is typically stored in ASCII or EBCDIC format although, other formats may be used with the present invention (i.e., for instance one of the many word processing formats). The plaintext representation of the software component is transformed by encryption function of cryptographic algorithm #1 303 using cryptographic key 305. Although a public key cryptographic algorithm may be used for cryptographic algorithm #1 303 it is preferable to use a conventional cryptographic algorithm like DES. This is due to performance considerations. Since the software component may be quite large, a faster encryption function is desired so that a reuser is not kept waiting during decryption. Also a conventional cryptographic algorithm makes browsing of the software component by the reuser much faster. Note that if a public key cryptographic algorithm type is used for the cryptographic algorithm #1 303 then the key supplied for cryptographic algorithm #2 313 would be the public key (assuming that the private key is used to encrypt).

The output of applying the encryption function for cryptographic algorithm #1 using the cryptographic key 305 is the encrypted software component 307. The encrypted software component 307 is then input to the hash function 309. The hash function 309 takes the encrypted software component 307 and produces a hash digest 311. Suitable hashing functions were discussed above. The hash digest 311 and the key 305 are then encrypted using a the encryption function of cryptographic algorithm #2 313. Cryptographic algorithm #2 313 must be of a public key type. Both the hash digest 311 and key 305 are encrypted using the encryption function of cryptographic algorithm # 2 with the private key 3 15. The private key must be properly safeguarded by the publisher. No one need know the private key 315 except the publisher. It is the public key associated with the private key, that can not be used to derive the private key, that is placed in the directory for use by reusers. The publisher must make the public key, associated with private key 315 available to potential reusers in order for the reusers to obtain the plaintext representation of the software component.

The hash digest 311 and key 305 are encrypted using the private key 315 and encryption function of cryptographic algorithm #2 313 to produce and encrypted hash digest 317 and the encrypted key 319. Therefore, we have produced the encrypted software component 307, encrypted hash digest 317, and the encrypted key 319. The only data component missing from that shown in FIG. 4 is the descriptive plaintext component. The descriptive plaintext component may be created by the publisher using any word processor or a by filling in a form provided by the reuse library or extracting the information from the plaintext representation of the software component or from design or requirements specifications or any other source. This information may consist of an abstract, a description, indexes, identification of other software component from which the particular software component was derived, identification of other software components required to make use of the software component, testing status, relationship to other software components, publisher identity, intellectual property information. The reuse library may provide this information or a portion of the information as well as additional information to the descriptive plaintext component. The descriptive plaintext component is discussed in detail below in the section discussing the reuse library.

FIG. 5 describes a method that can be implemented in hardware or software or any combination of hardware and software. In step 501 the plaintext representation of the software component is encrypted by a first cryptographic algorithm using a key. In step 503 the encrypted software component produced in step 501, is hashed to generate a hash digest. The hash digest from step 503 and the key used in step 501 are then encrypted using a second cryptographic algorithm in step 505. The second cryptographic algorithm being of a public key type, using the private key for encryption from the associated private and public keys. The encrypted software component of step 501, the encrypted hash digest of step 505 and the encrypted key of step 505 are then sent to the reuse library along with descriptive plaintext component for storage and other processing performed by the reuse library.

B. EMBODIMENTS

It should be noted that the functions described in FIG. 3 and steps of FIG. 5 may be carried out in either hardware or software or a combination of both. As mentioned in the cryptographic sections above many different cryptographic systems and functions are commercially available in hardware and software embodiments. Chip sets, boards, boxes, cards and software are available for performing the encrypting and hashing

functions required for publishing a software component. The preferred embodiment of the present invention is to have these functions performed using software so that interfacing with the reuse library can be performed from any computer system equipped with the software.

The preferred embodiment of the present invention comprises one or more software systems. In this context, software system is a collection of one or more executable software programs, and one or more storage areas, for example, RAM or disk. In general terms, a software system should be understood to comprise a fully functional software embodiment of a function, which can be added to an existing computer system to provide new function to that computer system. One embodiment of the present invention is shown in FIG. 8. The embodiment of the publishing workstation depicted in FIG. 8 is a collection of functions and data items. These functions and data items were explained in detail above. As shown in FIG. 8 the preferred embodiment of this invention comprises a set of computer programs for the generation of an encrypted software component 90, encrypted hash digest 91, encrypted key 92, from a software component 97 along with the descriptive plaintext component 93. FIG. 8 includes a processor 20 connected by means of a system bus 22 to a read only memory (ROM) 24 and memory 38. Also included in the computer system in FIG. 8 are a display 28 by which the computer presents information to the user, and a plurality of input devices including a keyboard 26, mouse 34 and other devices that may be attached via input/output port 30. Other input devices such as other pointing devices or a voice sensors or image sensors may also be attached. Other pointing devices include tablets, numeric keypads, touch screen, touch screen overlays, track balls, joy sticks, light pens, thumb wheels etc. The I/O 30 can be connected to communications lines, reuse library, directory, disk storage, input devices, output devices or other I/O equipment. The computer system shown in FIG. 8 may also be connected to the directory and reuse library via the communications adaptor 36. Communications between the publisher and other systems is provided via the communications manager 75. Communications manager 75 provides for the sending and receiving of data and requests. The memory 38 includes a display buffer 72 that contains pixel intensity values or character values for presentation on the display. The display 28 periodically reads the values from the display buffer 72 displaying these values or characters onto a display screen.

As shown in FIG. 8, the memory 38 includes a word processor 80, a hash function 81, an encryption function for cryptographic algorithm #1 82, a encryption function for cryptographic algorithm #2 83, hash digest 94, key 95 and private key 96.

The hash 81, encryption 82, encryption 83 and the word processor functions cause the software component record 89 with its four components: encrypted software component 90, encrypted hash digest 91, encrypted key 92 and the descriptive plaintext component 93 to be generated as described above. The supervisor 98 can coordinate the data flow between these functions and make sure the output generated is sent to the reuse library. Alternatively, each function can perform the required data flow as required. Also shown in the memory 38 is an operating system 74. Other elements shown in memory 38 include drivers 76 which interpret the electrical signals generated by devices such as the keyboard and mouse. A working memory area 78 is also shown in memory 38. The working memory area 78 can be utilized by any of the elements shown in memory 38. The working memory area can be utilized by any of functions it may also be used to store the various data items. The working memory area 78 may be partitioned amongst the elements and within an element. The working memory area 78 may be utilized for communication, buffering, temporary storage, or storage of data while a program is running.

V. REUSING

A. DESCRIPTION

The process used by the reuser is very similar to that used by the publisher except the steps are reversed. Referring now to FIG. 6 which depicts data objects as circles and functions as rectangles. FIG. 6 provides a functional overview of what a reuser needs to do in order to reuse a software component stored in the reuse library. Although not shown on FIG. 6 the reuser first retrieves or causes the reuse library to retrieve a software component record. The components of a software component record are depicted in FIG. 4. FIG. 6 shows the encrypted software component 601, the encrypted hash digest 603, and the encrypted key 605 of the retrieved software component record. Note that the reuser also requires public key 607. Public key 607 is obtained from the directory of publisher's public keys. The directory is discussed in detail below. The public key may be requested from the directory from information contained in the descriptive plaintext component (not shown) of the retrieved software component record. Thus the reuser knows which publisher's public key to request. The public key 607 is used with the decryption function of cryptographic algorithm #2 611. Using the public key 607, the decryption function processes the encrypted hash digest 603 and the encrypted key 605 to yield the hash digest 614 and the key 619, respectively.

The encrypted software component 601 is hashed by hash function 609 to yield hash digest 613. The hash function 609 utilized in the reuse function must be the same hash function that was utilized by the publisher. The hash digest 613 generated by the hash function 609 is then compared with the hash digest 614 decrypted from decryption function 611. This comparison is made by the comparator function 617. If the hash digest 613 and hash digest 614 are identical then no modification of the software component has taken place and no corruption will be indicated by the corruption indicator 623. If however, the hash digests are not identical then the software component has been corrupted and a corruption indication must be given. The indicator could be any visual or audible signal. A message flashing on the screen accompanied by beeping is usually sufficient to inform the reuser that the software component has been corrupted.

If no corruption indicator has been generated the encrypted software component is decrypted using the decryption function of algorithm #1 615 and the key 619 obtained from decryption function 611. Note even if a corruption indication was generated the encrypted software component might be decrypted but presumably the reuser would not want to use the corrupted component for fear of the effect of the corruption (e.g., possible viruses, bugs, etc.). The result of decryption function 615 using the encrypted software component

601 as input and the key 619 is the software component 621. The software component 621 can now be viewed by the reuser on the display, edited, modified or incorporated into a larger systems as the reuser desires. The reuser may also browse the software component 621 using the browsing means described below. If the comparator function 617 detects corruption it could tag (or place text inside) the plaintext representation of the software component 621 with a warning label that indicates to anyone browsing the software component that the component has been modified and that reuse is not recommended. FIG. 7 describes a method that can be implemented in hardware or software or any combination of hardware and software. In step 700 one or more software component records are retrieved or caused to be retrieved from the reuse library. In step 701 the public key associated with the publisher of the retrieved software component is obtained from a directory of publisher's public keys. In step 703 the encrypted hash digest and the encrypted key are decrypted using the second cryptographic algorithm and the public key obtained from step 701. In step 705 the encrypted software component is hashed into a hash digest. In step 707 the hash digest obtained from step 705 and the hash digest from step 703 are compared. If equal then an indication that the software component was not modified or corrupted may be provided (not shown). If not equal then in step 711 an indication is provided to the reuser that the software component has been corrupted or modified in some fashion. In step 713 the encrypted software component is decrypted using the first cryptographic algorithm and the key decrypted in step 703 to provide the plaintext representation of the software component. Other steps may be added for instance to support browsing as discussed below.

B. BROWSING MEANS

After retrieving a software component record from the reuse library the reuser may easily browse information contained in the descriptive plaintext component of the retrieved software component record. The browsing displays or prints information in human readable format. The browsing means can write directly to the display buffer or via operating system calls. The descriptive plaintext component of the software record contains information designed to enable the reuser to quickly determine whether a particular software component may be of value to the reuser. The reuser may also browse the plaintext representation of the software component. This requires that the encrypted software component of the software component record by decrypted, as described above. Because the present invention uses a symmetric cryptographic algorithm for encryption and decryption of the software component, decryption can be performed relatively fast. In fact, while the reuser is browsing the plaintext portion the decryption can be taking place in the background. Thus, the reuser is presented with information contained in the plaintext portion and then the plaintext representation of the software component in a seamless fashion so that decryption is transparent to the reuser. This requires an operating system that supports background processing or multi-processing or multi-tasking. If corruption of the software component is detected then the reuser could be informed as indicated above.

C. EMBODIMENT

It should be noted that the functions described in FIG. 6 and steps of FIG. 7 may be carried out in either hardware or software or a combination of both. As mentioned in the cryptographic sections above many different cryptographic systems and functions are commercially available in hardware and software embodiments. Chip sets, boards, boxes, cards and software are available for performing the decrypting and hashing functions required for reusing a software component or obtaining a plaintext representation of one. The preferred embodiment of the present invention is to have these functions performed using software so that interfacing with the reuse library can be performed from any computer system equipped with the software.

One embodiment of the present invention is shown in FIG. 9. The embodiment of the publishing workstation depicted in FIG. 9 is a collection of functions and data items. These functions and data items were explained in detail above. The preferred embodiment of the present invention comprises one or more software systems. In this context, software system is a collection of one or more executable software programs, and one or more storage areas, for example, RAM or disk. In general terms, a software system should be understood to comprise a fully functional software embodiment of a function, which can be added to an existing computer system to provide new function to that computer system. As shown in FIG. 9 the preferred embodiment of this invention comprises a set of computer programs for the generation of plaintext representation of the software component 97. The computer system of FIG. 9 includes a processor 20 connected by means of a system bus 22 to a read only memory (ROM) 24 and memory 38. Also included in the computer system in FIG. 9 are a display 28 by which the computer presents information to the reuser, and a plurality of input devices including a keyboard 26, mouse 34 and other devices that may be attached via input/output port 30. Other input devices such as other pointing devices or a voice sensors or image sensors may also be attached. Other pointing devices include tablets, numeric keypads, touch screen, touch screen overlays, track balls, joy sticks, fight pens, thumb wheels etc. The I/O 30 can be connected to communications lines, reuse library, directory, disk storage, input devices, output devices or other I/O equipment. The computer system shown in FIG. 9 may also be connected to the directory and reuse library via the communications adaptor 36. Communications between the reuser and other systems is provided via the communications manager 75. Communications manager 75 provides the for sending and receiving of data and requests. The memory 38 includes a display buffer 72 that contains pixel intensity values or character values for presentation on the display. The display 28 periodically reads the values from the display buffer 72 displaying these values or characters onto a display screen.

A software component record 89 obtained from the reuse library is shown in memory 38 along with its four components: the encrypted software 90, the encrypted hash digest 91, the encrypted key 92, and the descriptive plaintext component 93. A public key 901 is also shown in memory 38. The public key 901 may have been obtained via a network, via communications adapter 36 or input by the reuser using any of the input means specified above.

As shown in FIG. 9, the memory 38 includes a browsing means 903, a hash function 81, an decryption function for cryptographic algorithm #1 905, a decryption function for cryptographic algorithm #2 907, and a comparator function 909. These functions enable the computer system to obtain the plaintext representation of the software component 97 with an indication of whether or not the software component 97 has been corrupted as described above. The supervisor 911 can coordinate the data flow between these functions and make obtain the software component from the reuse library. The supervisor may after receiving the software component record 89 cause the browsing means 903 to immediately display information contained in the descriptive plaintext component 93 allowing the reuser to page or search through the information so provided. While permitting the reuser to browse, the encrypted software component 90 can be decrypted. As an alternative to the supervisor 911, each function can perform the required data flow as required.

Also shown in the memory 38 is an operating system 74. Other elements shown in memory 38 include drivers 76 which interpret the electrical signals generated by devices such as the keyboard and mouse. A working memory area 78 is also shown in memory 38. The working memory area 78 can be utilized by any of the elements shown in memory 38. The working memory area can be utilized by any of functions it may also be used to store the various data items. The working memory area 78 may be partitioned amongst the elements and within an element. The working memory area 78 may be utilized for communication, buffering, temporary storage, or storage of data while a program is running.

It should be noted that the reuser workstation and the publisher workstations embodiments can easily be provided in a single computer system that allows an operator to be a reuser and a publisher. This combined workstation may also contain an electronic directory.

VI. REUSE LIBRARY

A. DESCRIPTION

The reuse library is where the publishers send their software component record for access by the reusers. The reuse library can be electronically networked to the publishers or the publishers may simply send the encrypted software component through the mail on diskettes, tapes or other storage media. The reuse library must make the software component record available for browsing and selection by the reusers. Thus, the reuse library must provide storage and retrieval functions for the software component record. Each software component record received by the reuse library must be registered so that the publisher and software component record can be uniquely identified. This usually entails assigning the software component record a unique identifier.

In addition to the reuse library's storage and retrieval of software component records the reuse library may provide for other services as well. These include cataloguing so that software component records referencing other software component records can be easily located. Indexing and classifying the software components are other functions that the reuse library may provide to assist reuser's in more quickly and efficiently locating relevant software components. The reuse library may maintain its own classification data for classifying the software component. The reuse library may also require that the publisher fill out a requested form so that the classification may be more easily carried out. The classification criteria can consists of any criteria useful for discriminating among classes of reusers. Examples of classification criteria are domain or application oriented criteria. The reuse library may reclassify software components as the criteria evolve over time.

In order for the reuse library to perform these other functions, the publishers may be required to furnish certain plaintext information in the descriptive plaintext component of the software component record. The descriptive plaintext component thus contains information that is not encrypted. The information contained in the descriptive plaintext component need not be human readable but can be easily convened to human readable format. The information contained in the descriptive plaintext component may consist of an abstract, a description, indexes, identification of other software component from which the particular software component was derived, identification of other software components required to make use of the software component, testing status, relationship to other software components, publisher identity, intellectual property information. The abstract and/or description would among other things decide the context for which the software component was developed. Intellectual property information may contain licensing and/or derivation information and/or ownership information and/or a certificate of originality that certifies that the publisher created the work. Other software components referenced by the current software component may be identified by their unique identifier or some other suitable description. This information is essential for the reuse library to perform indexing, cataloguing and classification steps and to provide reusers with complete information on a particular software component in order to make a fully informed selection decision.

The reuse library may also add plaintext or a reference to plaintext that are reviews of the software components. The descriptive plaintext component could contain references to these reviews or actual contain the reviews themselves. Reusers may provide reviews concerning their experiences with using or adapting the software component for their own purposes. The reviews may also be created by independent reviewers as the market place for reusable components grows. In summary the descriptive plaintext component may contain information provided from any source. The descriptive plaintext component may be assembled by the reuse library or the publisher or both.

FIG. 4 depicts the logical view of a record for one software component 401 that is provided to the reuse library. The reuse library in registering the software component record would assign the software component record a unique identifier (not shown) and ! hen perform the necessary functions to store the software component record. The software component record 401 consists of four components: the encrypted software component 403, the encrypted hash digest 405, the encrypted key 407 and the descriptive plaintext component 409. Note the descriptive plaintext component is not the plaintext representation of the software component. As was stated above the descriptive plaintext component 409 may consist of any or all of the following data: abstract, a description, indexes, identification of other software component from which the particular software component was derived, identification of other software components required to make use of the

software component, testing status, relationship to other software components, publisher identity, intellectual property information. Other information may also be included in the descriptive plaintext component as required or needed.

It should be noted that the software component record 401 as shown in FIG. 4 is a logical view of the software component record. The software component record 401 may be physically stored in a variety of manners. Thus, the software components may be stored as a flat file or in database table or set of database tables or as objects or sets of objects in an object oriented database. A repository may also be used for storing the information contained in the software component record. (See IBM System Journal Repository Manager Technology, Vol. 29, No. 2, 1990 pp 209–227, by Sagawa hereby incorporated by reference). It should be noted that any method of storing and retrieving the information contained in the software component record will work with the present invention. However, database or repository embodiments are preferred because they permit the reusers to more easily search and locate a desired software component and makes the reuse library easier to maintain. This searching/browsing may entail decryption of the encrypted software component or may use the plaintext description and references as described above.

The reuse library may be located on any computer system with suitable storage capability. A file server or database server machine where access is provided to publishers and reusers via a client/server architecture is a preferred embodiment. The publishers and reusers may be connected by phone lines, LAN, WAN, MAN, wireless, cellular telephone or any other communications means.

As was mentioned above the present invention contemplates working with multiple libraries. The libraries may specialize in particular problem domains, particular languages, any other criteria or combinations of the above.

B. EMBODIMENTS

One embodiment for the reuse library of the present invention is shown in FIG. 9. The embodiment of the reuse library as depicted in FIG. 9 is a collection of functions and data items. These functions data items were explained in detail above.

The preferred embodiment of the present invention comprises one or more software systems. In this context, software system is a collection of one or more executable software programs, and one or more storage areas, for example, RAM or disk. In general terms, a software system should be understood to comprise a fully functional software embodiment of a function, which can be added to an existing computer system to provide new function to that computer system. As shown in FIG. 10 the preferred embodiment of this invention comprises a set of computer programs for the storage and retrieval of software component records.

The computer system of FIG. 10 includes a processor 20 connected by means of a system bus 22 to a read only memory (ROM) 24 and memory 38. Also included in the computer system in FIG. 9 are a display. 28 by which the computer presents information to the operator of the reuse library and a plurality of input devices including a keyboard 26, mouse 34 and other devices that may be attached via input/output port 30. The I/O 30 can be connected to communications lines, publisher's workstations, reuser's workstations, other reuse libraries, directory, disk storage, input devices, output devices or other I/O equipment. The computer system shown in FIG. 10 may also be connected to the publisher's workstations and the reuser's workstations via the communications adaptor 36. Communications between the reuse library and other systems is provided via the communications manager 75. Communications manager 75 provides the for sending and receiving of data and requests. The memory 38 includes a display buffer 72. The display 28 periodically reads the values from the display buffer 72 displaying these values or characters onto a display screen.

A plurality of software component records 1001 are shown in memory 38 each having four components: the encrypted software, the encrypted hash digest, the encrypted key, and the descriptive plaintext component. The software components may be stored in a repository, a relational database, a flatfile, object oriented data base or any other means. The retrieval means 1003 and the storage means 1005 would then interface with the storage subsystem for the storage and retrieval of software component records. Also shown in memory are cataloging means 1007, classifying means 1009, and indexing means 1011. Also shown are the classifying criteria 1013, software component reviews 1015 and indexes 1017. The retrieval means 1005 may also contain a search capability that permits reusers connected to the reuse library to search through software component records 1001 uses a search criteria, key words, classification criteria, etc.. This search capability could use information contained in the descriptive plaintext component of the software component records.

Also shown in the memory 38 is an operating system 74. Other elements shown in memory 38 include drivers 76 which interpret the electrical signals generated by devices such as the keyboard and mouse. A working memory area 78 is also shown in memory 38. The working memory area 78 can be utilized by any of the elements shown in memory 38. The working memory area can be utilized by any of functions it may also be used to store the various data items. The working memory area 78 may be partitioned amongst the elements and within an element. The working memory area 78 may be utilized for communication, buffering, temporary storage, or storage of data while a program is running.

It should be noted that the reuser workstation and the publisher workstations and reuse library embodiments can easily be provided in a single computer system that allows an operator to be a reuser and a publisher or a librarian. This combined workstation may also contain an electronic directory.

VII. DIRECTORY

The directory of publisher's public keys 105 (FIG 1 & FIG. 2) is basically a list or table. One column in the table contains the publisher and the other column contains the publisher's public key. The publisher's public key is required by the reuser in order to obtain the plaintext representation of any software component the publisher places in the reuse library. A publisher may have more than one public key (this implies that the publisher has more than one private keys). Publishers require write access to the directory or the ability to place their public keys in the reuse library.

The reuser desiring to obtain or browse a software component must obtain the publisher public key. The directory may be contained on the same computer sys-

tem as the reuse library or another computer system or the reuser's computer system or no system at all. Thus the public keys may be contained in a notebook or print out. In this case the public key would be looked up in the book for input to the decryption hardware or software. Reusers require read access to the directory.

If the directory is stored on a computer system the reuser would obtain the public key and use it to decrypt the desired software component as described above. The reuser may obtain the public key by requesting the key associated with the publisher described in the plaintext portion of the software component record or via the unique identifier or a table provided by the reuse library or some other means.

The directory may be placed on any computer system. A file server or database server where access is provided to publishers and reusers via a client/server architecture is a preferred embodiment. The publishers and reusers may be connected by phone lines, LAN, WAN, MAN, wireless, cellular network or other communications means.

The directory may also be placed in a trusted system with access to the directory provided via a trusted path. Using a trusted system and a trusted path provides additional security in that only authorized individuals would have access to the public keys. Since the public keys are added to the directory by a trusted path and are obtained by only those parties granted access to the trusted system containing the directory would be able to access the directory and the public keys. A trusted file server can be utilized to provide an additional security mechanism. The trusted file server essential keeps the publisher's public keys semi-private in that only those individuals who are provided access to the trusted system can obtain access to the public keys.

The present invention may also be utilized with a certification management system. A certification management system can provide for the directory required by the present invention. A certification management system would handle public keys for other purposes and may also provided a means for certifying electronic signatures as well. The certification management system might be part of a greater encryption infrastructure. The certification management system may allow others to electronicly look up each other's public keys. The certification management system could also handle key exchanges and digital signatures. The reusers and publishers might be connected to such a system by Interact or other network.

VIII. ADVANTAGES AND CLOSING

This present invention provides several advantages. The first advantage is that large software components are encrypted and decrypted using a fast private key scheme (like DES) rather than the slow public key methods. A second advantage is that the software component is sent to the reuse library and retrieved from the reuse library in encrypted form so that reusers cannot ignore authentication requirements. A third advantage is that the key associated with the conventional algorithm (the DES key) is encrypted so that adversaries cannot simply substitute a replacement key to accompany replacement text. A fourth advantage is that the present invention can work any of the current cryptographic standards, like DES, and potential standards, like skipjack. Alternatively, the present invention can utilize the current de facto standard, RSA, or other public key methods rather than RSA.

While the invention has been described in detail herein in accord with certain preferred embodiments thereof, modifications and changes therein may be effected by those skilled in the art. Accordingly, it is intended by the appended claims to cover all such modifications and changes as fall within the true spirit and scope of the invention.

What is claimed:

1. In a network of computers comprising at least one computer, the method for reusing software components that maintains the integrity and authenticity of the software components, said method comprising:

generating an software component record using the following substeps:

(a) encrypting a plaintext representation of a software component into a encrypted software component with a first cryptographic algorithm using first key;

(b) hashing the encrypted software component to generate a first hash digest;

(c) encrypting the first hash digest and the first key using a second cryptographic algorithm with a second key, wherein said second cryptographic algorithm is of a public key type and said second key is the private key associated with at least one public key, said software component record consisting of the encrypted software component, the encrypted hash digest, and the encrypted first key; storing the software component record in a reuse library;

retrieving the software component record from the reuse library;

generating the plaintext representation of the software component using the following substeps:

(a) obtaining a public key associated with the second key from a public key directory;

(b) decrypting the encrypted hash digest and the encrypted first key into the decrypted first key and the decrypted first hash digest using the public key and the second cryptographic algorithm;

(c) hashing the encrypted software component to generate a second hash digest;

(d) comparing the second hash digest with the decrypted first hash digest, and if not identical indicating that the software component is corrupted, if identical indicating that the software is not corrupted;

(e) decrypting the encrypted software component into the plaintext representation using the decrypted first key and the first encryption algorithm.

2. The method of claim 1 wherein the Data Encryption Standard is used as the first cryptographic algorithm and the Digital Signature Algorithm is used as the second cryptographic algorithm.

3. The method of claim 1 wherein the Data Encryption Standard is used as the first cryptographic algorithm and the RSA is used as the second cryptographic algorithm.

4. The method of claim 1 wherein the Skipjack is used as the first cryptographic algorithm and the Digital Signature Algorithm is used as the second cryptographic algorithm.

5. The method of claim 1 wherein the Skipjack is used as the first cryptographic algorithm and the RSA is used as the second cryptographic algorithm.

6. The method of claim 1 wherein the software component record includes a descriptive plaintext component containing a description of the software component.

7. The method of claim 6 wherein the descriptive plaintext component includes information regarding data rights and ownership rights.

8. The method of claim 6 wherein the plaintext representation of the software component also includes information defining the software components relationship to other software component records in the reuse library.

9. The method of claim 7 wherein the plaintext representation of the software component also includes information defining the software components relationship to other software component records in the reuse library.

10. The computer system comprising:

a reuse library having a plurality of encrypted software components each software component record having an encrypted software component, an encrypted hash digest, and an encrypted first key;

said reuse library having a storage means for storing encrypted software components;

said reuse library having a retrieval means for retrieving encrypted software components;

a directory containing a list of publishers and an associated list of public keys;

at least one publisher's workstation coupled to the reuse library, said publishers workstation having

an first encrypting means for encrypting a plaintext representation of a software component into a encrypted software component with a first cryptographic algorithm using first key;

an hashing means for hashing the encrypted software component to generate a first hash digest;

a second encrypting means for encrypting the first hash digest and the first key using a second cryptographic algorithm with a second key, wherein said second cryptographic algorithm is of a public key algorithm type and said second key is the publisher's private key associated with a publisher's public key, said software component record consisting of the encrypted software component, the encrypted hash digest, and the encrypted first key;

a communications means for sending the software component record to the reuse library for storage by the storage means;

at least one reuser workstation coupled to the reuse library, said reuser workstation having

a requesting means for sending a request to the reuse library for a desired encrypted software component, wherein said request causes the retrieval means of the reuse library to retrieve the desired software component and send it to the requesting workstation;

a means for obtaining the public key from the directory, said public key associated with the second key of the desired encrypted software component;

a first decrypting means for decrypting the encrypted hash digest and the encrypted first key into the decrypted first key and the decrypted first hash digest using the public key and the second cryptographic algorithm;

an hashing means for hashing the encrypted software component to generate a second hash digest;

a comparing means for comparing the second hash digest with the decrypted first hash digest, and if not identical indicating that the software component is corrupted, if identical indicating that the software is not corrupted;

a second decrypting means for decrypting the encrypted software component into the plaintext representation using the decrypted first key and the first encryption algorithm.

11. The system of claim 10 wherein the reuser workstation also includes a display means for displaying the plaintext representation of the software component record and for providing an indication of corruption.

12. The system of claim 10 wherein the reuser workstation also includes a browsing means for browsing encrypted software components stored in the reuse library.

13. The system of claim 10 wherein the plurality of encrypted software components and the directory are stored in a relational database.

14. The system of claim 10 wherein the plurality of encrypted software components and the directory are stored in a object oriented database.

15. The system of claim 10 wherein the reuse library includes a catalogue means for assigning the software component record a unique identifier and for classifying the software component record.

16. The system of claim 15 wherein the catalogue means classifies the software component record according to a set of classification criteria.

17. The system of claim 10 wherein the Data Encryption Standard is used as the first cryptographic algorithm and the Digital Signature Algorithm is used as the second cryptographic algorithm.

18. The system of claim 10 wherein the Data Encryption Standard is used as the first cryptographic algorithm and the RSA is used as the second cryptographic algorithm.

19. The system of claim 10 wherein the Skipjack is used as the first cryptographic algorithm and the Digital Signature Algorithm is used as the second cryptographic algorithm.

20. The system of claim 10 wherein the Skipjack is used as the first cryptographic algorithm and the RSA is used as the second cryptographic algorithm.

* * * * *

UNITED STATES PATENT AND TRADEMARK OFFICE

CERTIFICATE OF CORRECTION

PATENT NO. : 5,343,527

DATED : August 30, 1994

INVENTOR(S) : James W. Moore

It is certified that error appears in the above-identified patent and that said Letters Patent is hereby corrected as shown below:

Title page, before item [57] Abstract, in the "Attorney, Agent or Firm" "Mark A. Wurm" to —John D. Flynn, Mark A. Wurm—.

Column 1, Line 17, change "theft" to —their— (2 occurrences).

In Claim 1, Column 20, Line 13, change "an" to —a—.

In Claim 1, Column 20, line 17, change "a" to —an—.

In Claim 8, Column 21, line 10, change "components" to —component's—.

In Claim 9, Column 21, line 15, change "components" to —component's—.

In Claim 10, Column 21, line 31, change "an first" to —a first—.

In Claim 10, Column 21, line 32, change "into a" to —into an—.

In Claim 10, Column 21, line 35, change "an hashing means" to —a hashing means—.

In Claim 10, Column 22, line 9, change "an hashing means" to —a hashing means—.

UNITED STATES PATENT AND TRADEMARK OFFICE

CERTIFICATE OF CORRECTION

PATENT NO. : 5,343,527

DATED : August 30, 1994

INVENTOR(S) : James W. Moore

It is certified that error appears in the above-indentified patent and that said Letters Patent is hereby corrected as shown below:

In Claim 14, Column 22, line 33, change "a object oriented" to —an object oriented—.

Signed and Sealed this

Twenty-second Day of November, 1994

Attest:

BRUCE LEHMAN

Attesting Officer

Commissioner of Patents and Trademarks

United States Patent [19]

Kawamura et al.

[11] **Patent Number: 5,046,094**

[45] **Date of Patent: Sep. 3, 1991**

[54] **SERVER-AIDED COMPUTATION METHOD AND DISTRIBUTED INFORMATION PROCESSING UNIT**

[75] Inventors: **Shinichi Kawamura; Atsushi Shimbo; Kyoto Takabayashi, all of Yokohama, Japan**

[73] Assignee: **Kabushiki Kaisha Toshiba, Kawasaki, Japan**

[21] Appl. No.: **474,404**

[22] Filed: **Feb. 2, 1990**

[30] **Foreign Application Priority Data**

Feb. 2, 1989	[JP]	Japan	1-24723
May 18, 1989	[JP]	Japan	1-122849
May 31, 1989	[JP]	Japan	1-139593
Jul. 20, 1989	[JP]	Japan	1-189677
Sep. 21, 1989	[JP]	Japan	1-243349
Dec. 15, 1989	[JP]	Japan	1-323926

[51] Int. Cl.[5] **H04L 9/02**

[52] U.S. Cl. **380/46; 380/28; 380/23**

[58] Field of Search 380/23-25, 380/28, 30, 44, 46; 340/825.31, 825.34

[56] **References Cited**

U.S. PATENT DOCUMENTS

4,351,982	9/1982	Miller et al.	380/25
4,633,036	12/1986	Hellman et al.	380/25
4,797,920	1/1989	Stein	380/24
4,933,970	6/1990	Shamir	380/23 X
4,969,189	11/1990	Ohta et al.	380/25

OTHER PUBLICATIONS

"A Method For Obtaining Digital Signatures and Public-Key Cryptosystems"; Ronald L. Rivest, Adi Shamir, Len Adleman; (Apr. 4, 1977).

"New Direction In Cryptography"; Whitfield Diffie, Martine E. Hellman; IEEE Transactions on Information Theory, vol. IT-22, No. 6, Nov. 1976.

"Smart Cards Can Compute Secret Heavy Functions with Powerful Terminals"; T. Matsumoto, K. Kato, H. Imai; The 10th Symposium on Information Theory and Its Applications, Nov. 19-21, 1987.

"Speeding Up Secret Computations with Insecure Auxiliary Devices"; T. Matsumoto, K. Kato, H. Imai; Pro. of Crypto '88, Aug. 21-25, 1988.

"On Using RSA With Low Exponent In A Publish Key Network"; J. Hastad, pp. 403-408.

"Fast Decipherment Algorithm For RSA Public-Key Cryptosystem"; *Electronics Letter*, Oct. 14, 1982, vol. 18, No. 21.

"How to Ask Services Without Violating Privacy", T. Matsumoto, H. Imai; The 1989 Symposium on Cryptography and Information Security, Feb. 2-4, 1989.

"Computation Methods for RSA with the Aid of Powerful Terminals"; S. Kawamura, A. Shimbo; The 1989 Symposium on Cryptography and Information Security, Feb. 2-4, 1989.

"On Asking Advices with Privacy"; K. Kato, T. Matsumoto, H. Imai; 1988 Symposium on Cryptography & Information Security, Feb. 22-24, 1988.

"A Method for Obtaining Digital Signatures & Public-Key Cryptosystem", R. Rivest, A. Shamir, L. Adleman; Communications of the ACM, Feb. 1978, vol. 21, No. 2, pp. 120-126.

A Japanese Publication, pp. 18-19.

Primary Examiner—Thomas H. Tarcza
Assistant Examiner—Tod Swann
Attorney, Agent, or Firm—Oblon, Spivak, McClelland, Maier & Neustadt

[57] **ABSTRACT**

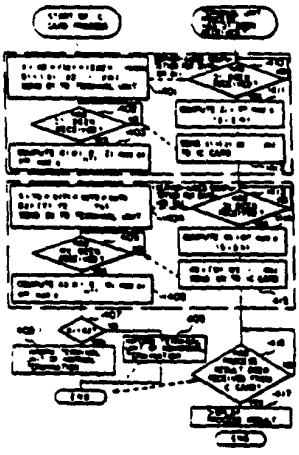
A server-aided computation method using a main unit for processing secret information and at least one auxiliary unit for supporting a computation that said main unit executes, said method comprising the steps of generating d' from a secret key d' using m random numbers R_i (where $i=1, \ldots, m$) generated by said main unit having secret keys n and d, transferring d' and n from said main unit to said auxiliary unit, computing the following equation from a message block C in said auxiliary unit

$$M = C^{d'} \bmod n$$

computing X using said random numbers R_i and n in said main unit while computing M' in said auxiliary unit, transferring M' from said auxiliary unit to said main unit, and computing a message block M using the following equation in said main unit.

$$M = M' \cdot X \bmod n$$

12 Claims, 21 Drawing Sheets



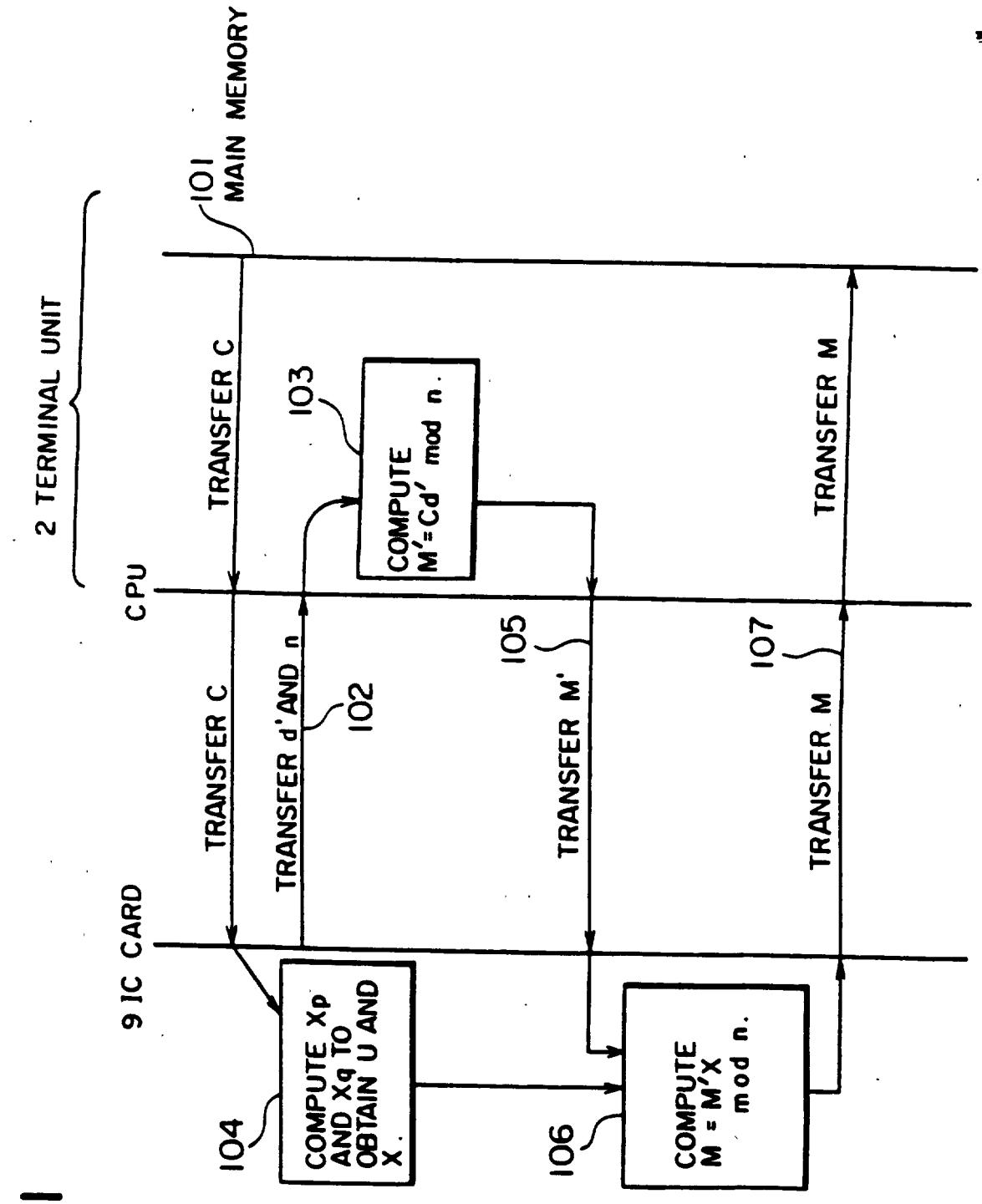

2 TERMINAL UNIT
101 MAIN MEMORY
TRANSFER C
103
COMPUTE M'=Cd' mod n.
TRANSFER M
CPU
TRANSFER C
TRANSFER d'AND n
102
105
TRANSFER M'
107
TRANSFER M
9 IC CARD
104
COMPUTE Xp AND Xq TO OBTAIN U AND X.
106
COMPUTE M = M'X mod n.

FIG. 1

FIG. 2

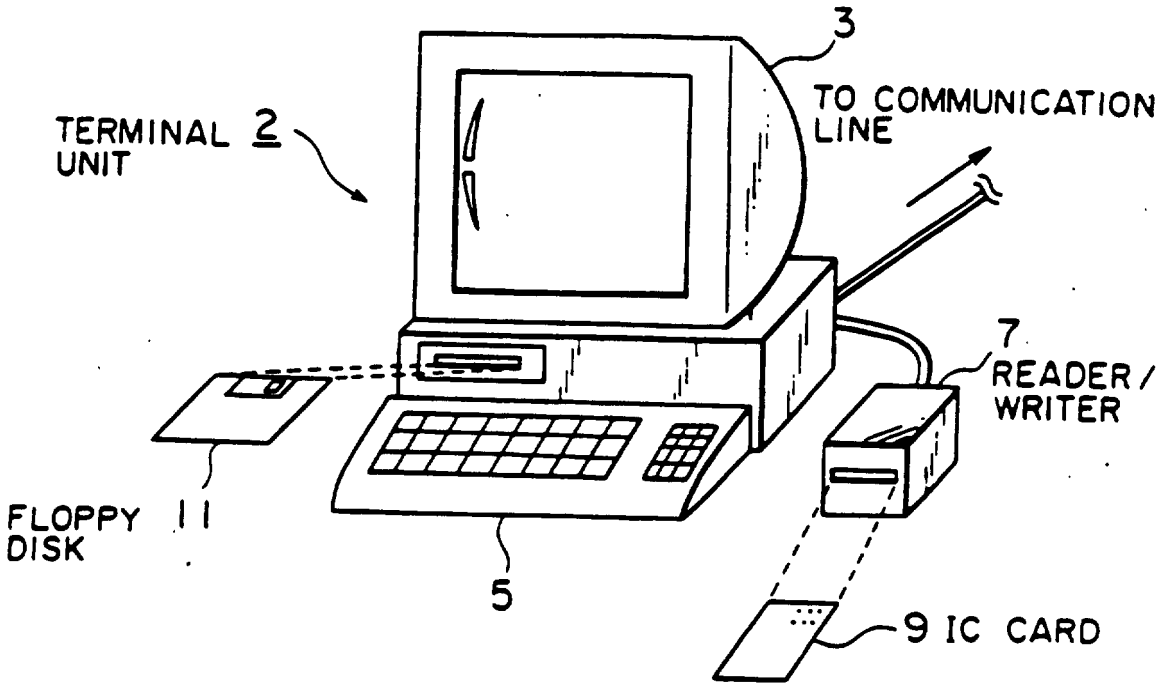

3
TO COMMUNICATION LINE
TERMINAL 2 UNIT
7
READER / WRITER
FLOPPY 11 DISK
5
9 IC CARD

FIG. 3

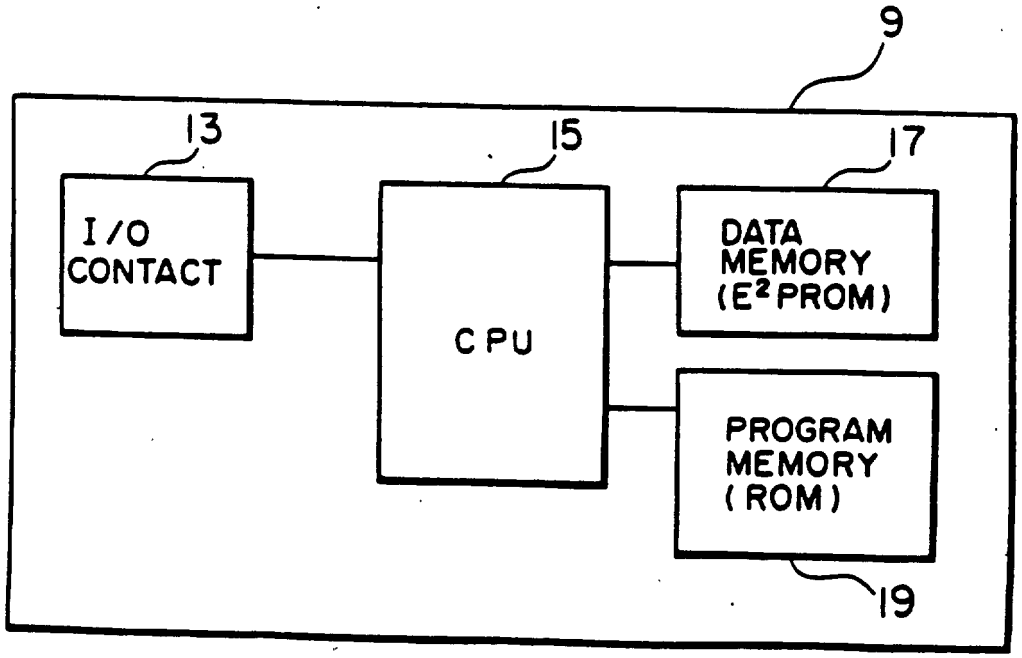

9
13
15
17
I/O CONTACT
CPU
DATA MEMORY (E²PROM)
PROGRAM MEMORY (ROM)
19

FIG. 4

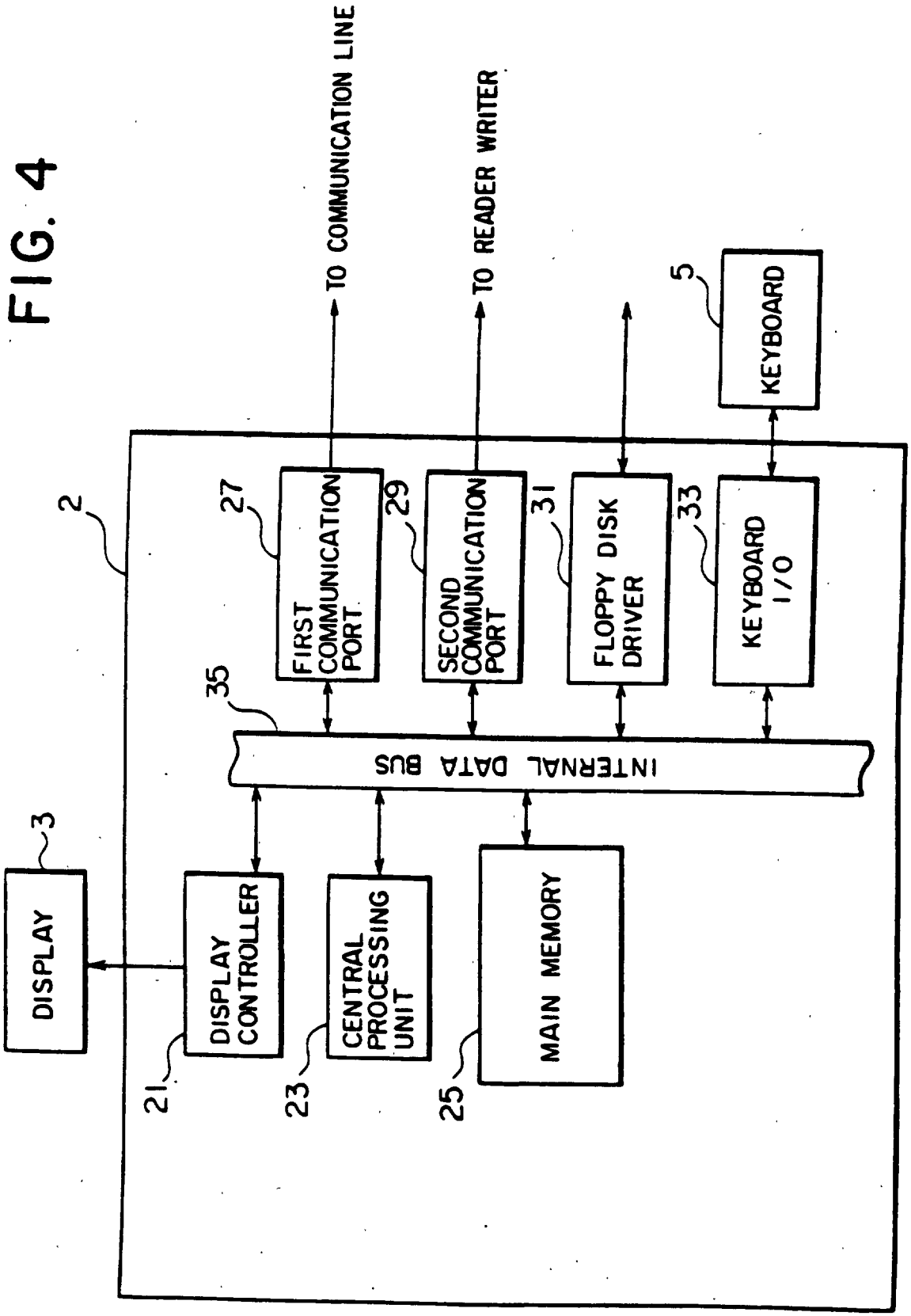
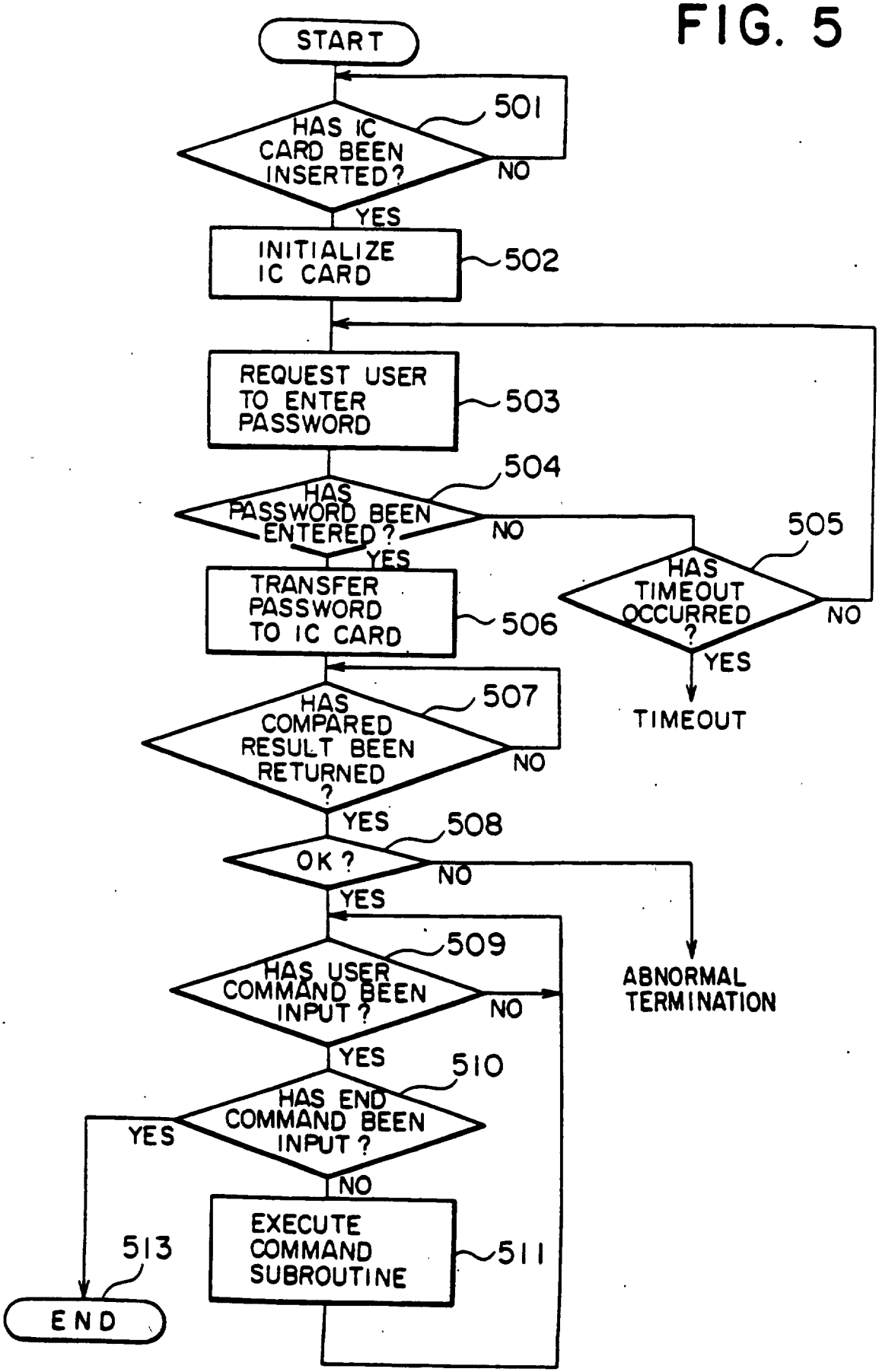

TO COMMUNICATION LINE
TO READER WRITER
5
KEYBOARD
2
27
FIRST COMMUNICATION PORT
29
SECOND COMMUNICATION PORT
31
FLOPPY DISK DRIVER
33
KEYBOARD I/O
35
INTERNAL DATA BUS
3
DISPLAY
DISPLAY CONTROLLER
21
CENTRAL PROCESSING UNIT
23
MAIN MEMORY
25

FIG. 5


START
HAS IC CARD BEEN INSERTED?
501
NO
YES
INITIALIZE IC CARD
502
REQUEST USER TO ENTER PASSWORD
503
HAS PASSWORD BEEN ENTERED?
504
NO
YES
TRANSFER PASSWORD TO IC CARD
506
HAS TIMEOUT OCCURRED ?
505
NO
YES
TIMEOUT
HAS COMPARED RESULT BEEN RETURNED ?
507
NO
YES
OK ?
508
NO
YES
ABNORMAL TERMINATION
HAS USER COMMAND BEEN INPUT ?
509
NO
YES
HAS END COMMAND BEEN INPUT ?
510
YES
NO
EXECUTE COMMAND SUBROUTINE
511
513
END

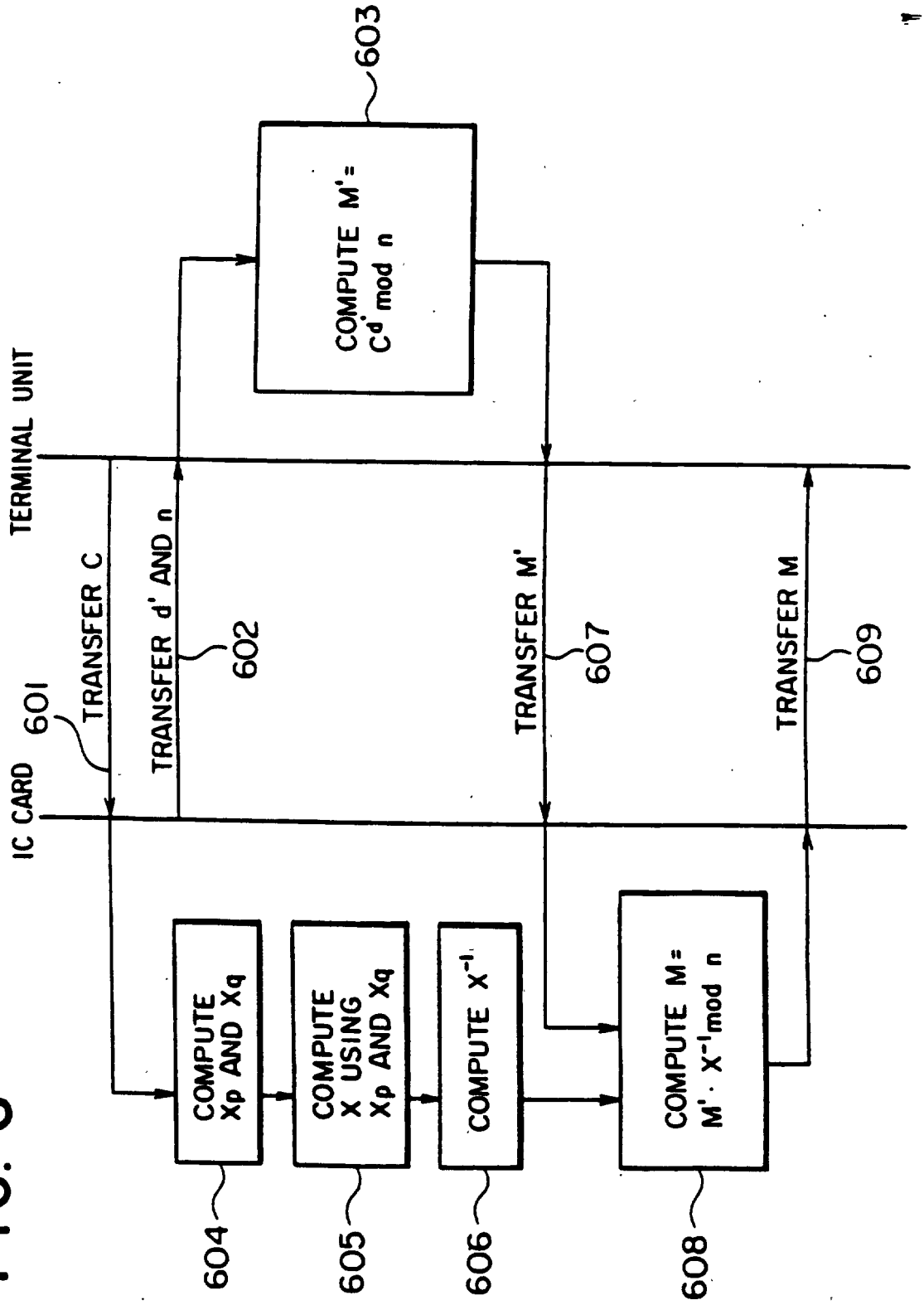

IC CARD
TERMINAL UNIT
601
TRANSFER C
TRANSFER d' AND n
602
COMPUTE M'= C^d' mod n
603
TRANSFER M'
607
TRANSFER M
609
COMPUTE Xp AND Xq
604
COMPUTE X USING Xp AND Xq
605
COMPUTE X^-1
606
COMPUTE M= M'·X^-1 mod n
608

FIG. 6

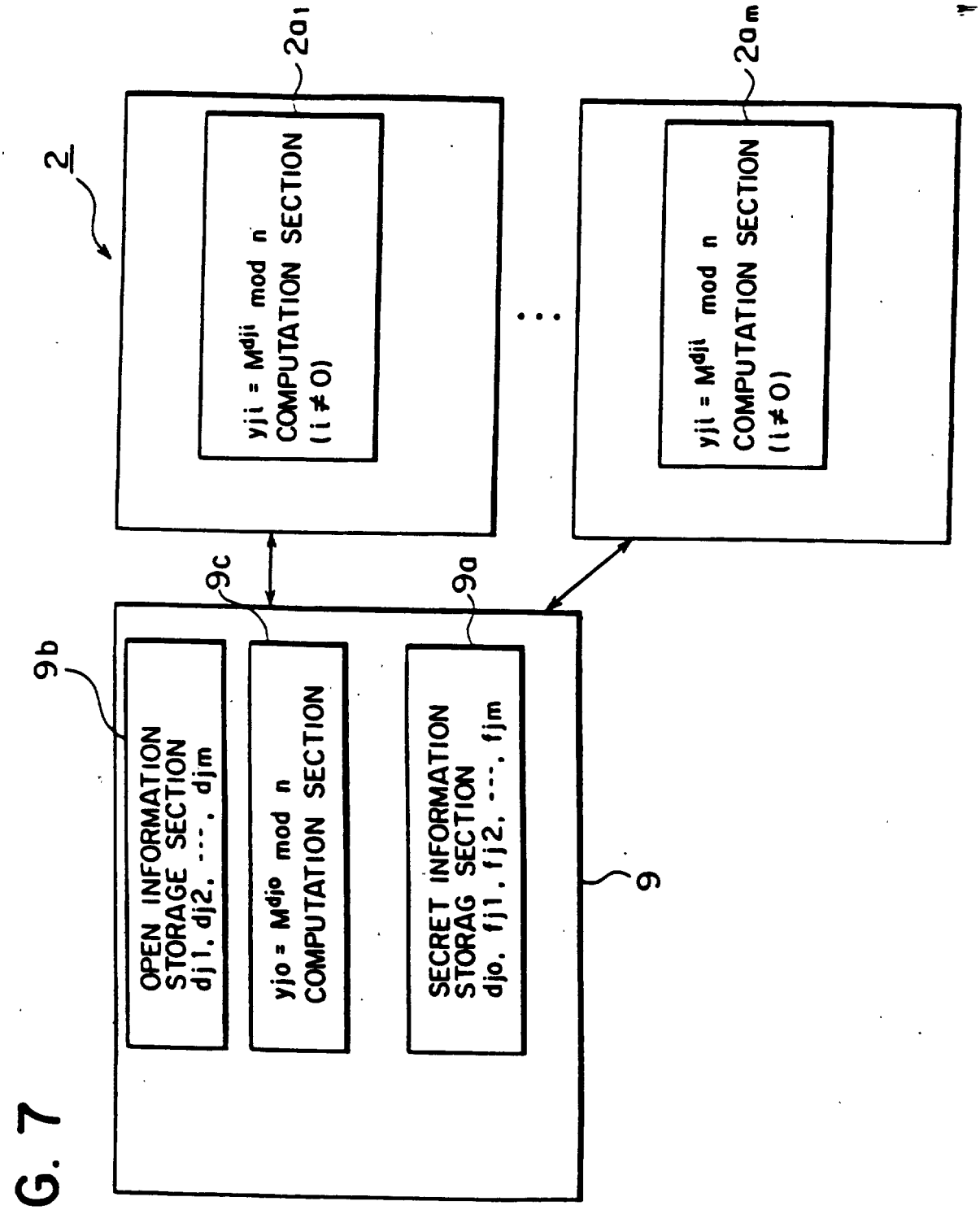

2
2a1
yji = M^dji mod n
COMPUTATION SECTION
(i ≠ 0)
2am
yji = M^dji mod n
COMPUTATION SECTION
(i ≠ 0)
9c
9a
9b
OPEN INFORMATION
STORAGE SECTION
dj1, dj2, ---, djm
yjo = M^djo mod n
COMPUTATION SECTION
SECRET INFORMATION
STORAG SECTION
djo, fj1, fj2, ---, fjm
9

FIG. 7

FIG. 8

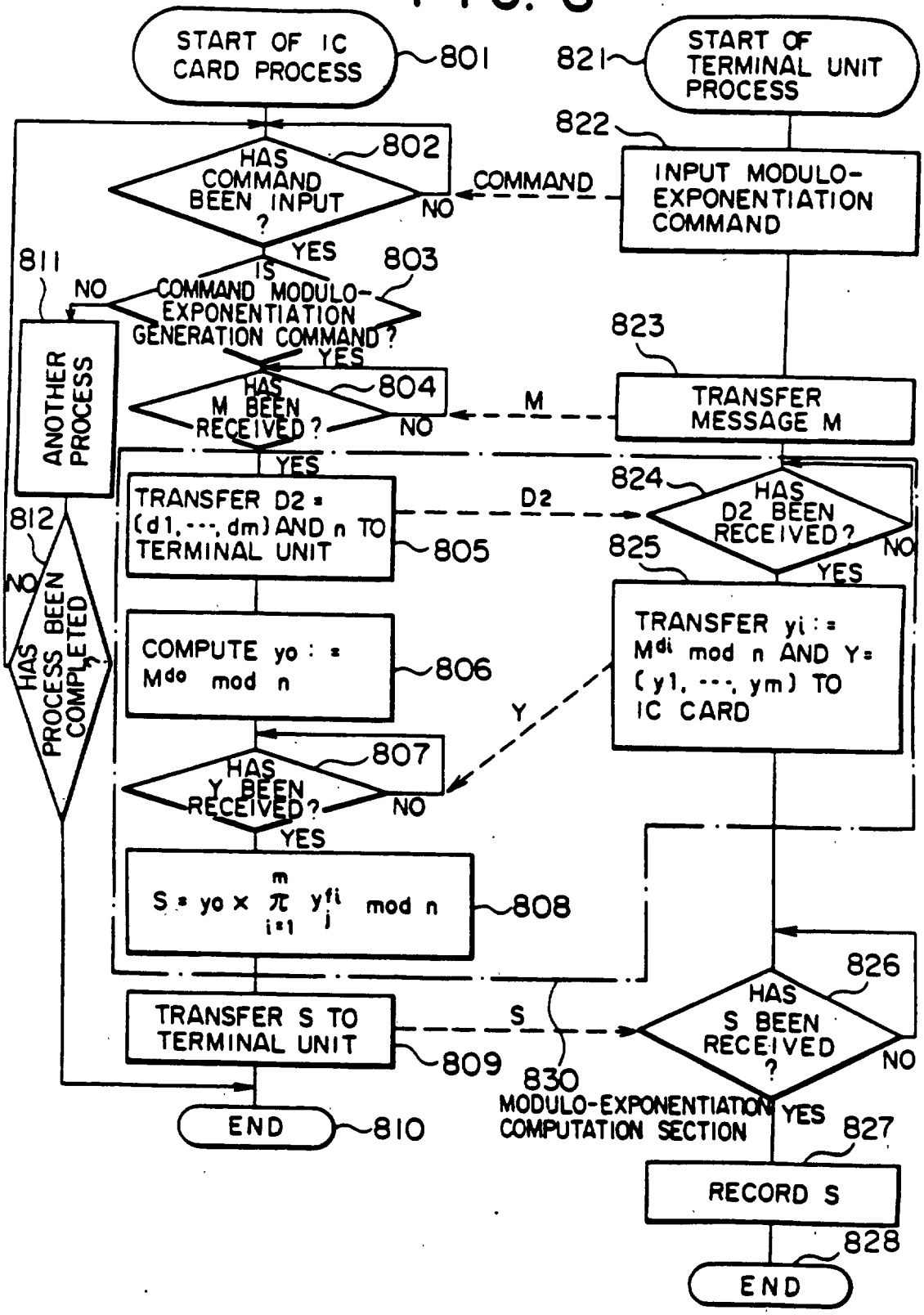

START OF IC CARD PROCESS
801
HAS COMMAND BEEN INPUT ?
802
NO
YES
COMMAND
IS COMMAND MODULO-EXPONENTIATION GENERATION COMMAND ?
803
NO
YES
ANOTHER PROCESS
811
HAS M BEEN RECEIVED ?
804
NO
YES
M
TRANSFER D2 = (d1, ---, dm) AND n TO TERMINAL UNIT
805
D2
COMPUTE yo : = M^do mod n
806
HAS Y BEEN RECEIVED ?
807
NO
YES
Y
S = yo × π (i=1 to m) y_i^fi mod n
808
TRANSFER S TO TERMINAL UNIT
809
S
END
810
HAS PROCESS BEEN COMPLETED ?
812
NO
START OF TERMINAL UNIT PROCESS
821
INPUT MODULO-EXPONENTIATION COMMAND
822
TRANSFER MESSAGE M
823
HAS D2 BEEN RECEIVED ?
824
NO
YES
TRANSFER yi : = M^di mod n AND Y = (y1, ---, ym) TO IC CARD
825
HAS S BEEN RECEIVED ?
826
NO
YES
RECORD S
827
END
828
MODULO-EXPONENTIATION COMPUTATION SECTION
830

FIG. 9

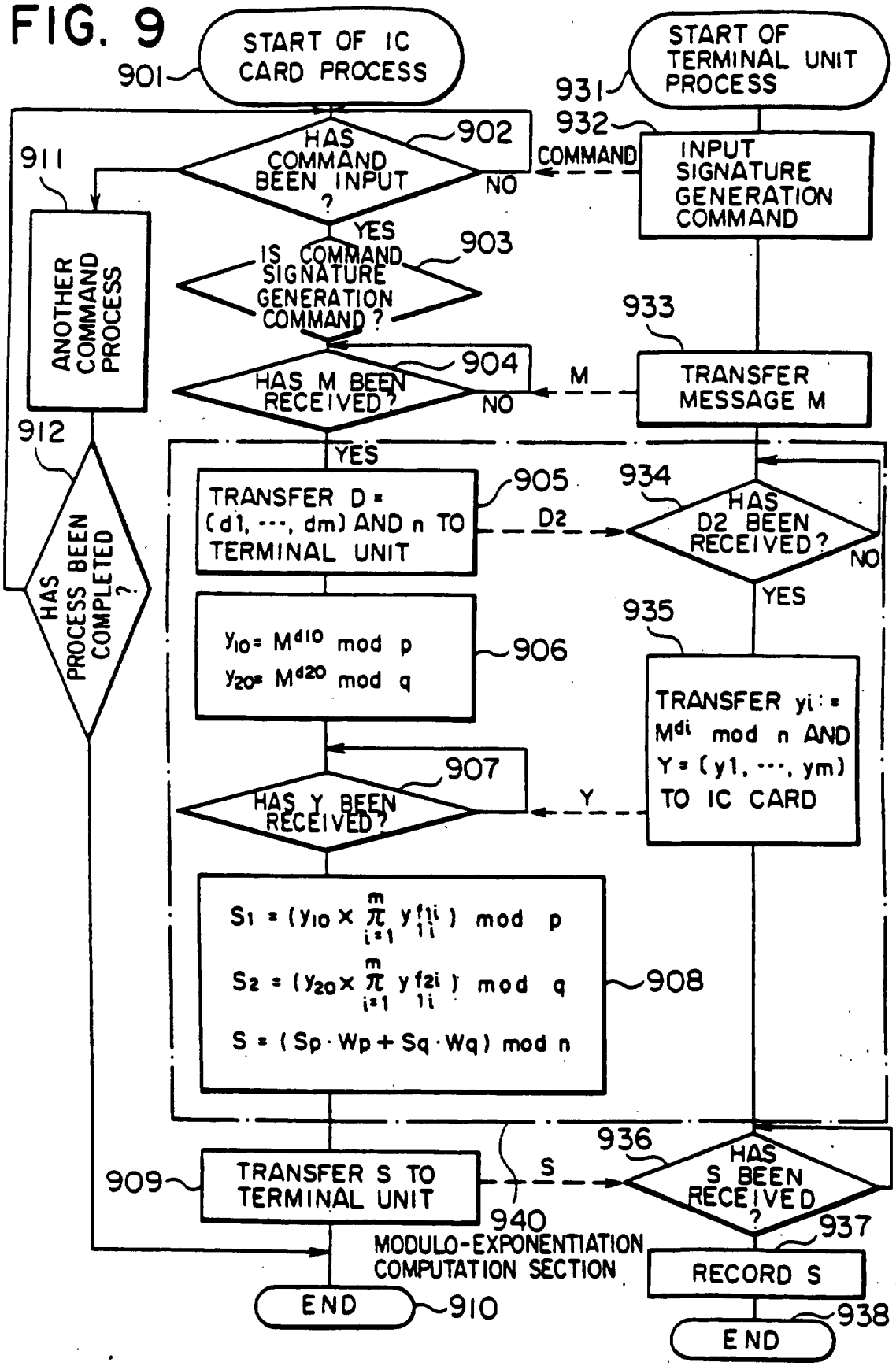

START OF IC CARD PROCESS
901
START OF TERMINAL UNIT PROCESS
931
HAS COMMAND BEEN INPUT ?
902
NO
YES
COMMAND
932
INPUT SIGNATURE GENERATION COMMAND
911
ANOTHER COMMAND PROCESS
IS COMMAND SIGNATURE GENERATION COMMAND ?
903
933
HAS M BEEN RECEIVED?
904
M
NO
TRANSFER MESSAGE M
912
YES
HAS PROCESS BEEN COMPLETED ?
TRANSFER D = (d1, ---, dm) AND n TO TERMINAL UNIT
905
D2
934
HAS D2 BEEN RECEIVED?
NO
YES
$y_{10} = M^{d10} \bmod p$
$y_{20} = M^{d20} \bmod q$
906
935
TRANSFER yi = $M^{di} \bmod n$ AND Y = (y1, ---, ym) TO IC CARD
HAS Y BEEN RECEIVED?
907
Y
$S_1 = (y_{10} \times \prod_{i=1}^{m} y_{1i}^{f1i}) \bmod p$
$S_2 = (y_{20} \times \prod_{i=1}^{m} y_{1i}^{f2i}) \bmod q$
$S = (Sp \cdot Wp + Sq \cdot Wq) \bmod n$
908
936
HAS S BEEN RECEIVED ?
TRANSFER S TO TERMINAL UNIT
909
S
940
MODULO-EXPONENTIATION COMPUTATION SECTION
937
RECORD S
END
910
938
END

FIG. 10

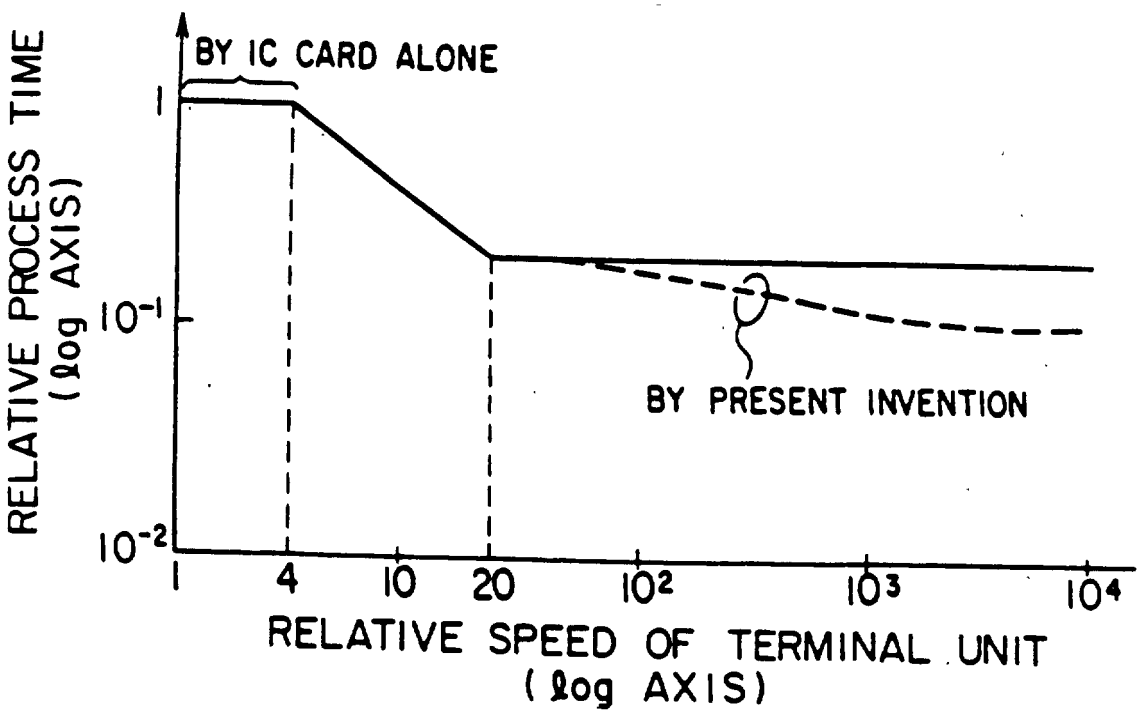

BY IC CARD ALONE
RELATIVE PROCESS TIME
(log AXIS)
1
10^{-1}
10^{-2}
BY PRESENT INVENTION
1
4
10
20
10^2
10^3
10^4
RELATIVE SPEED OF TERMINAL UNIT
(log AXIS)

FIG. 11

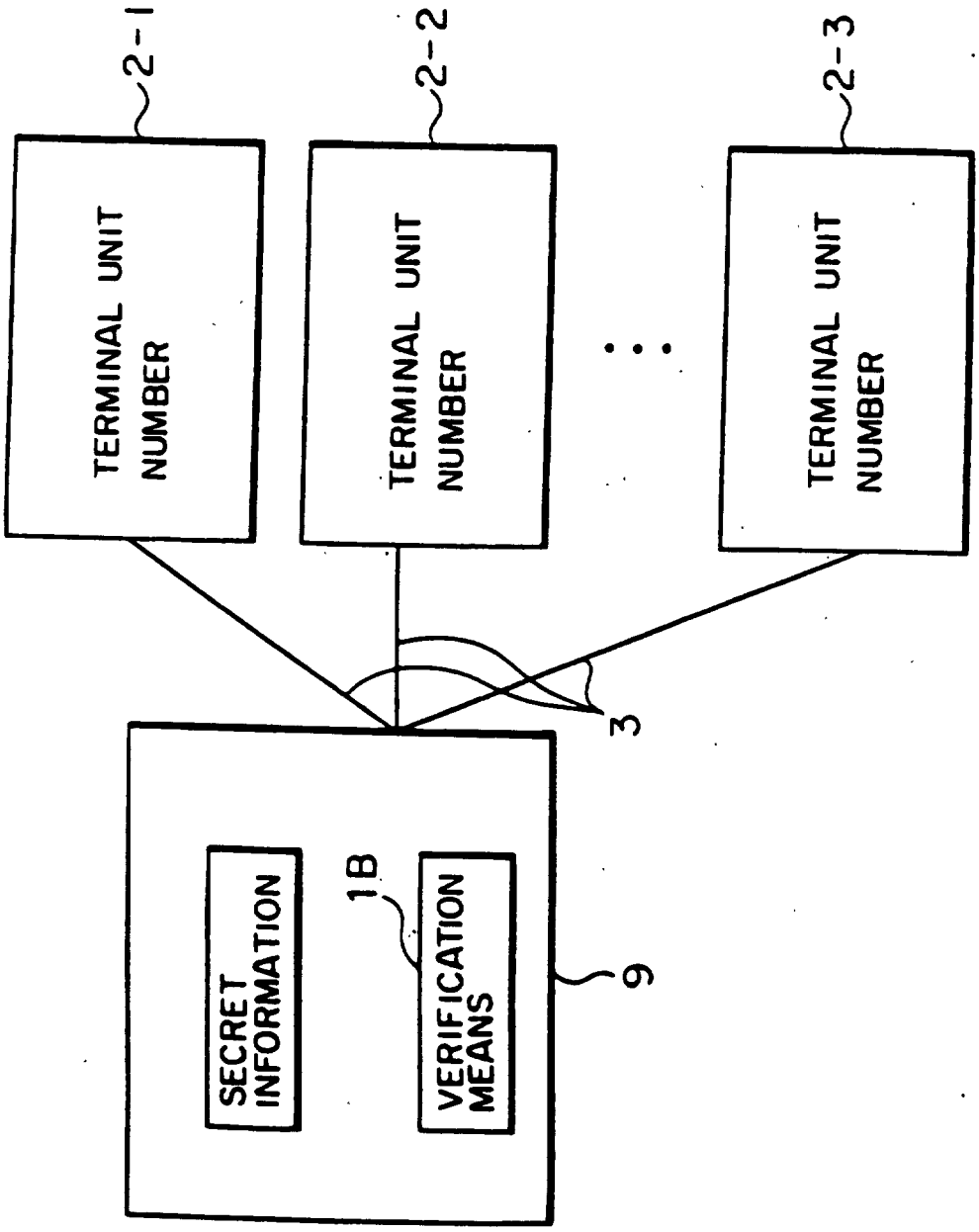
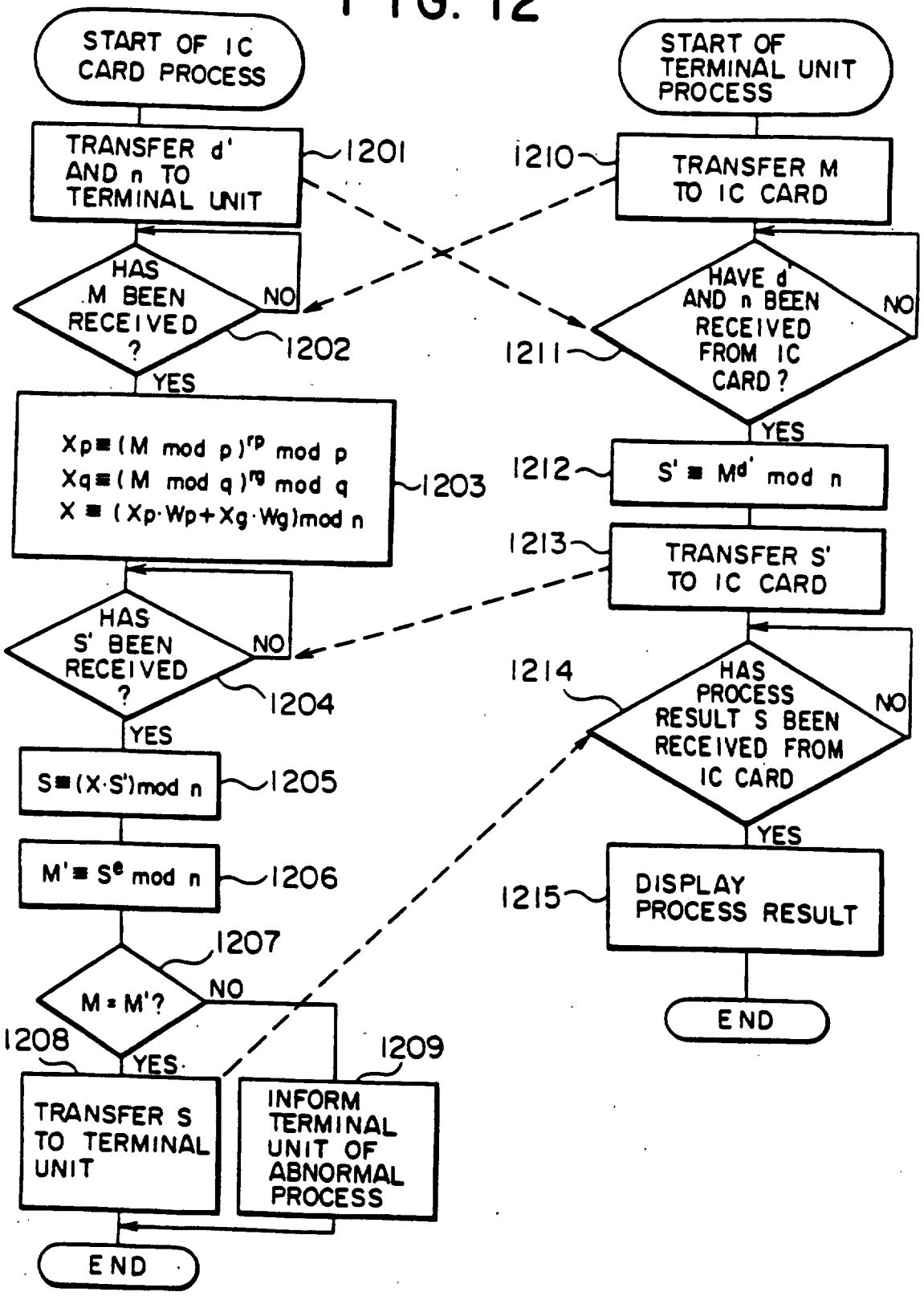

2-1
2-2
2-3
TERMINAL UNIT NUMBER
TERMINAL UNIT NUMBER
TERMINAL UNIT NUMBER
3
SECRET INFORMATION
1B
VERIFICATION MEANS
9

FIG. 12


START OF IC CARD PROCESS
TRANSFER d' AND n TO TERMINAL UNIT
1201
HAS M BEEN RECEIVED ?
NO
1202
YES
Xp≡(M mod p)^rp mod p
Xq≡(M mod q)^rq mod q
X ≡ (Xp·Wp+Xq·Wq)mod n
1203
HAS S' BEEN RECEIVED ?
NO
1204
YES
S≡(X·S')mod n
1205
M'≡S^e mod n
1206
1207
M=M'?
NO
1208
YES
1209
TRANSFER S TO TERMINAL UNIT
INFORM TERMINAL UNIT OF ABNORMAL PROCESS
END
START OF TERMINAL UNIT PROCESS
1210
TRANSFER M TO IC CARD
HAVE d' AND n BEEN RECEIVED FROM IC CARD ?
NO
1211
YES
1212
S'≡M^d' mod n
1213
TRANSFER S' TO IC CARD
1214
HAS PROCESS RESULT S BEEN RECEIVED FROM IC CARD
NO
YES
1215
DISPLAY PROCESS RESULT
END

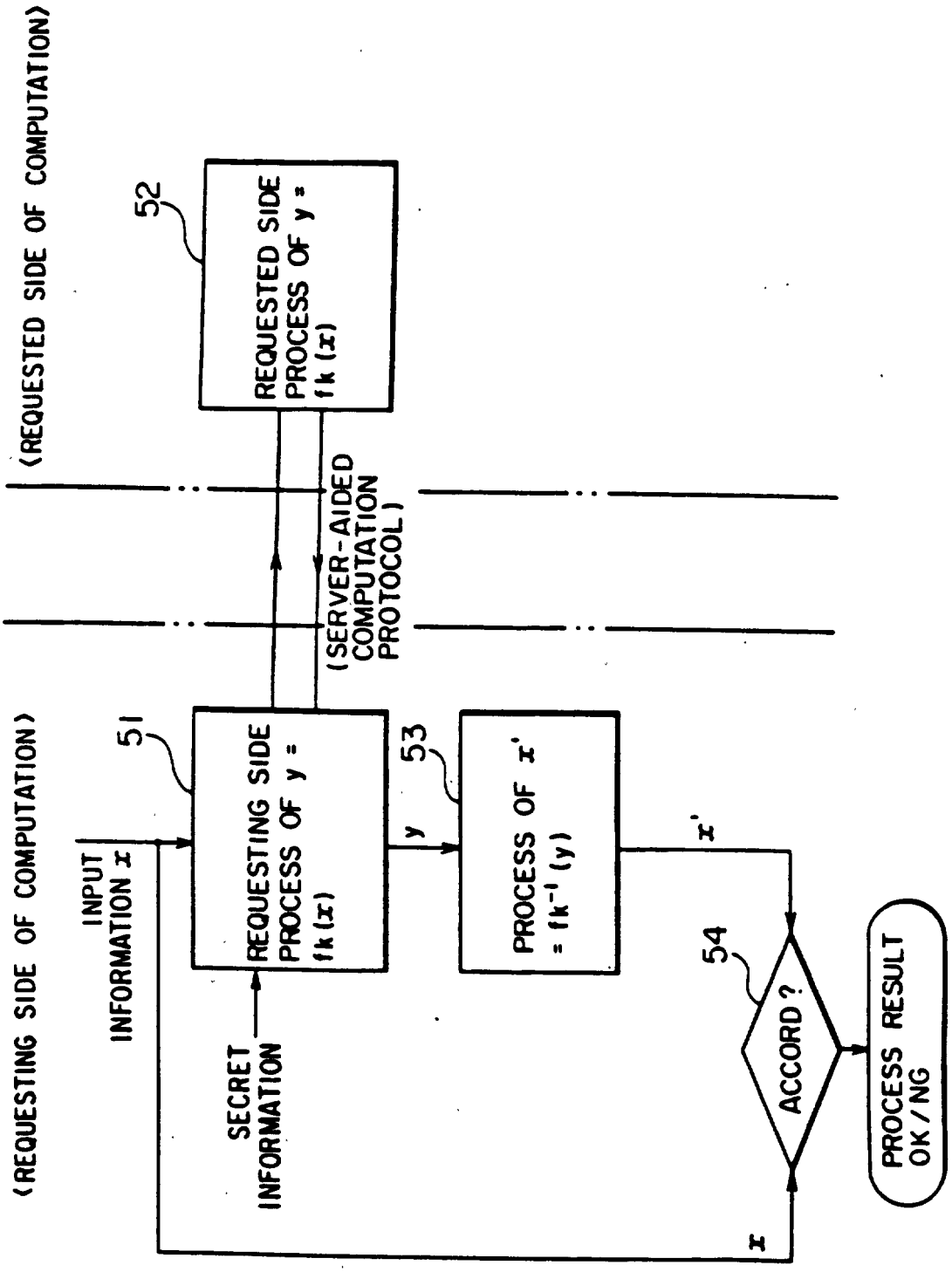
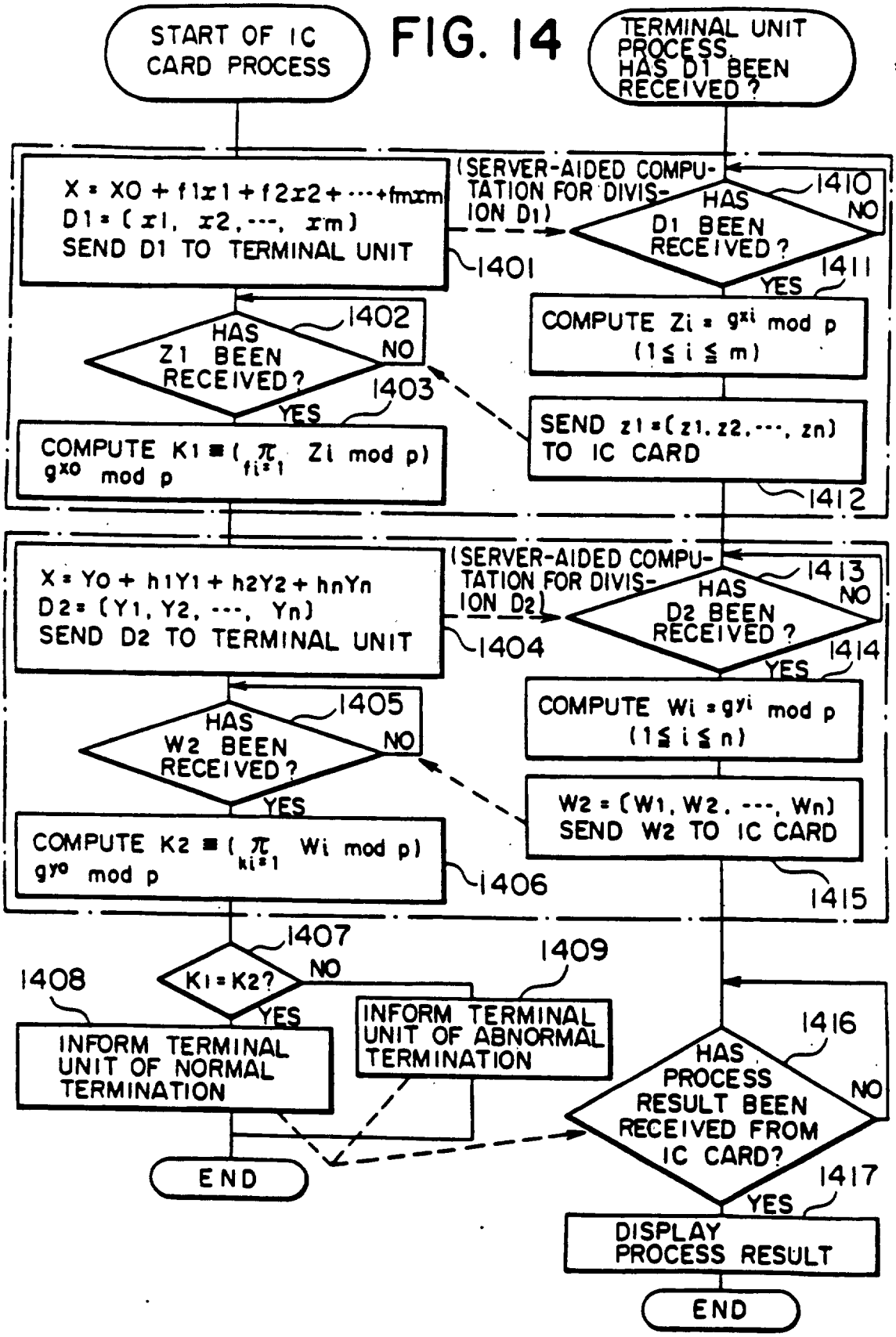

(REQUESTED SIDE OF COMPUTATION)
52
REQUESTED SIDE PROCESS OF y = fk(x)
(SERVER-AIDED COMPUTATION PROTOCOL)
(REQUESTING SIDE OF COMPUTATION)
51
INPUT INFORMATION x
REQUESTING SIDE PROCESS OF y = fk(x)
SECRET INFORMATION
y
53
PROCESS OF x' = fk⁻¹(y)
x'
54
ACCORD?
PROCESS RESULT OK/NG
x

FIG. 13

FIG. 14


START OF IC CARD PROCESS
TERMINAL UNIT PROCESS, HAS D1 BEEN RECEIVED?
X = X0 + f1x1 + f2x2 + ···+fmxm
D1 = (x1, x2, ···, xm)
SEND D1 TO TERMINAL UNIT
1401
(SERVER-AIDED COMPUTATION FOR DIVISION D1)
HAS D1 BEEN RECEIVED?
1410
NO
YES
1411
COMPUTE Zi = g^xi mod p (1≦i≦m)
HAS Z1 BEEN RECEIVED?
1402
NO
YES
1403
SEND z1 = (z1, z2, ···, zn) TO IC CARD
1412
COMPUTE K1 ≡ (π fi=1 Zi mod p) g^x0 mod p
X = Y0 + h1Y1 + h2Y2 + hnYn
D2 = (Y1, Y2, ···, Yn)
SEND D2 TO TERMINAL UNIT
1404
(SERVER-AIDED COMPUTATION FOR DIVISION D2)
HAS D2 BEEN RECEIVED?
1413
NO
YES
1414
COMPUTE Wi = g^yi mod p (1≦i≦n)
HAS W2 BEEN RECEIVED?
1405
NO
YES
W2 = (W1, W2, ···, Wn) SEND W2 TO IC CARD
1415
COMPUTE K2 ≡ (π ki=1 Wi mod p) g^y0 mod p
1406
K1 = K2?
1407
NO
YES
1408
1409
INFORM TERMINAL UNIT OF NORMAL TERMINATION
INFORM TERMINAL UNIT OF ABNORMAL TERMINATION
HAS PROCESS RESULT BEEN RECEIVED FROM IC CARD?
1416
NO
YES
1417
DISPLAY PROCESS RESULT
END
END

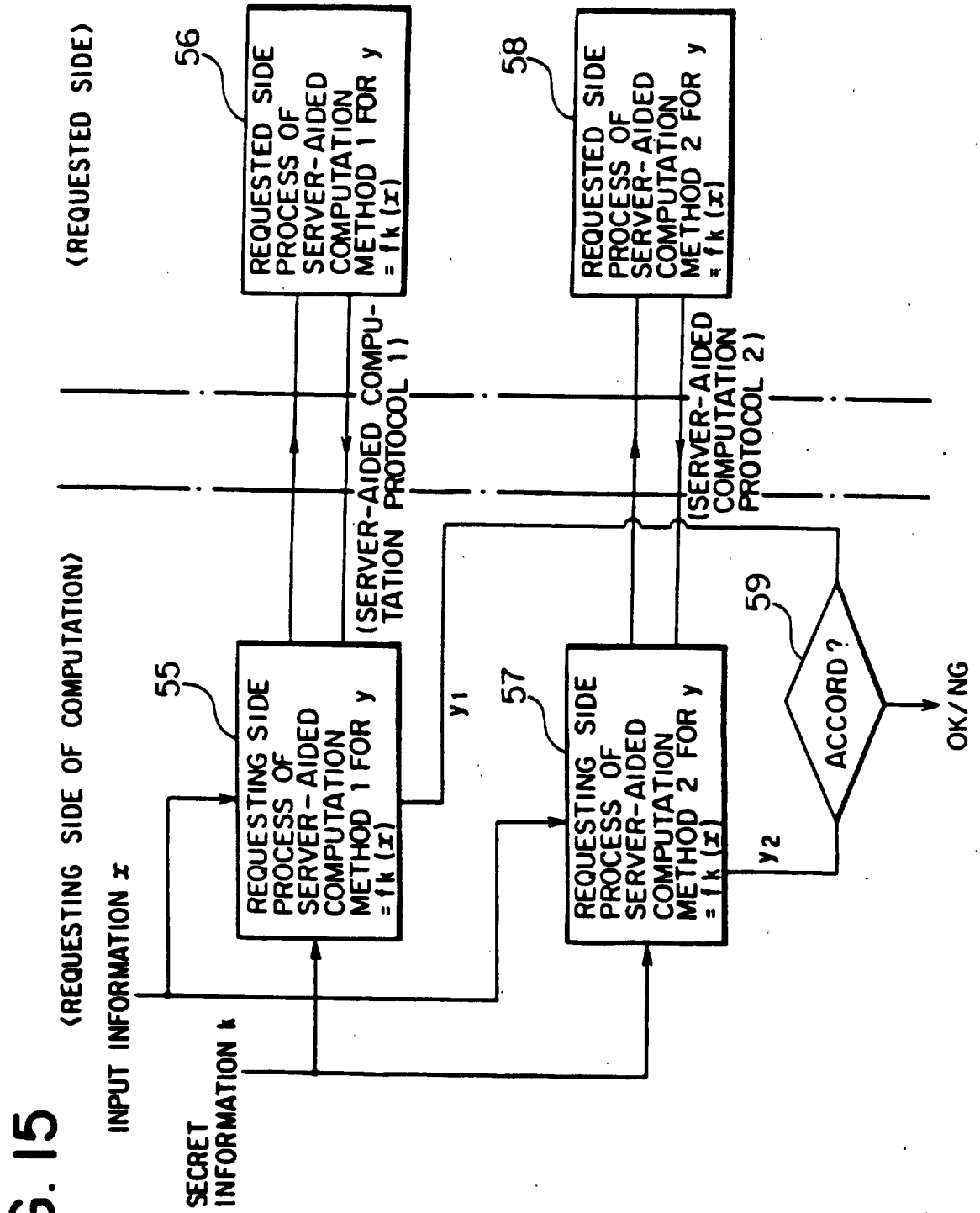

(REQUESTED SIDE)
(REQUESTING SIDE OF COMPUTATION)
INPUT INFORMATION x
SECRET INFORMATION k
56
REQUESTED SIDE PROCESS OF SERVER-AIDED COMPUTATION METHOD 1 FOR y = fk (x)
58
REQUESTED SIDE PROCESS OF SERVER-AIDED COMPUTATION METHOD 2 FOR y = fk (x)
(SERVER-AIDED COMPUTATION PROTOCOL 1)
(SERVER-AIDED COMPUTATION PROTOCOL 2)
55
REQUESTING SIDE PROCESS OF SERVER-AIDED COMPUTATION METHOD 1 FOR y = fk (x)
y1
57
REQUESTING SIDE PROCESS OF SERVER-AIDED COMPUTATION METHOD 2 FOR y = fk (x)
y2
59
ACCORD?
OK/NG

FIG. 15

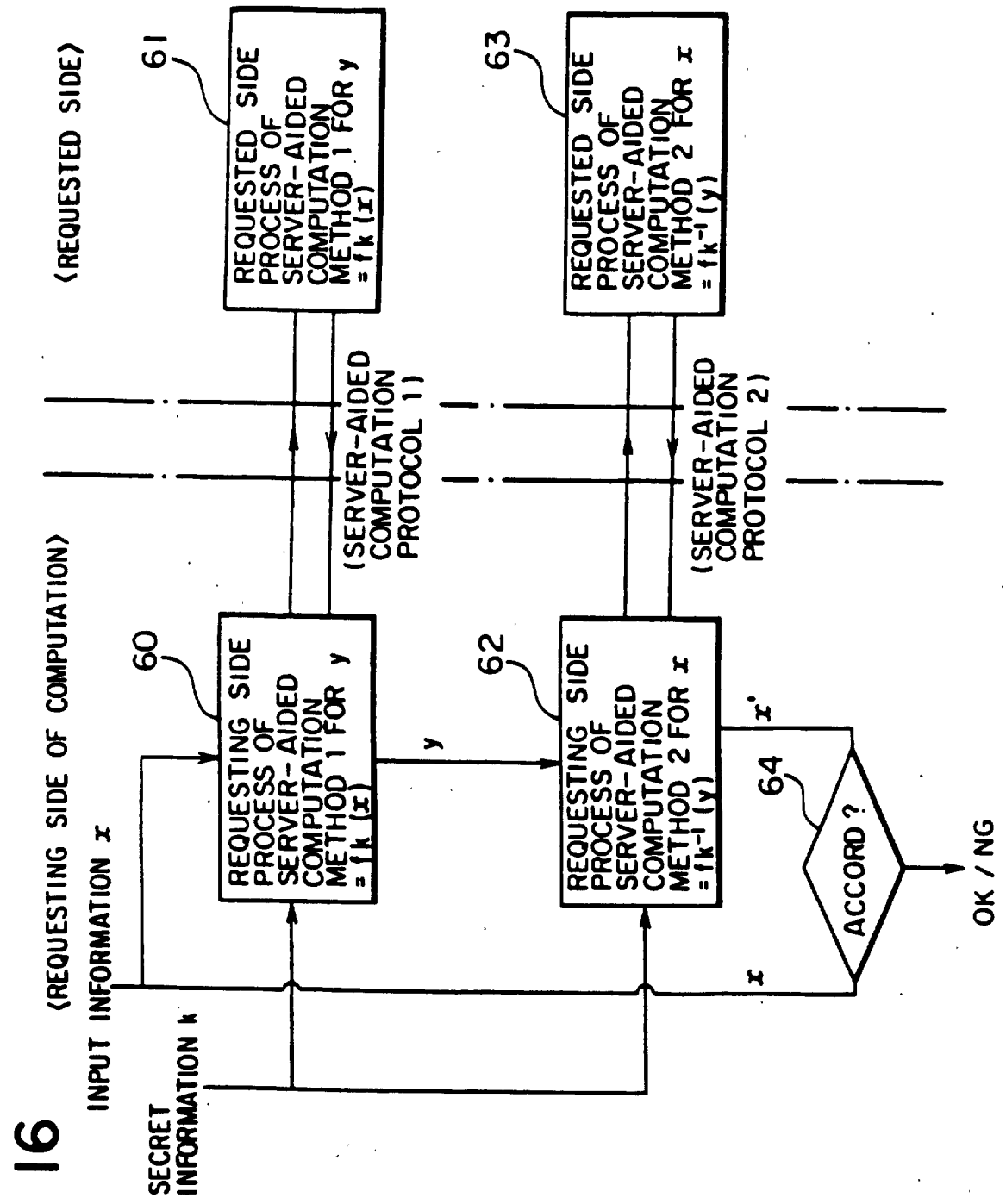

(REQUESTED SIDE)
61
REQUESTED SIDE PROCESS OF SERVER-AIDED COMPUTATION METHOD 1 FOR y = fk (x)
63
REQUESTED SIDE PROCESS OF SERVER-AIDED COMPUTATION METHOD 2 FOR x = fk⁻¹ (y)
(SERVER-AIDED COMPUTATION PROTOCOL 1)
(SERVER-AIDED COMPUTATION PROTOCOL 2)
(REQUESTING SIDE OF COMPUTATION)
60
REQUESTING SIDE PROCESS OF SERVER-AIDED COMPUTATION METHOD 1 FOR y = fk (x)
62
REQUESTING SIDE PROCESS OF SERVER-AIDED COMPUTATION METHOD 2 FOR x = fk⁻¹ (y)
y
x'
64
ACCORD ?
OK / NG
INPUT INFORMATION x
SECRET INFORMATION k
x

FIG. 16

FIG. 17

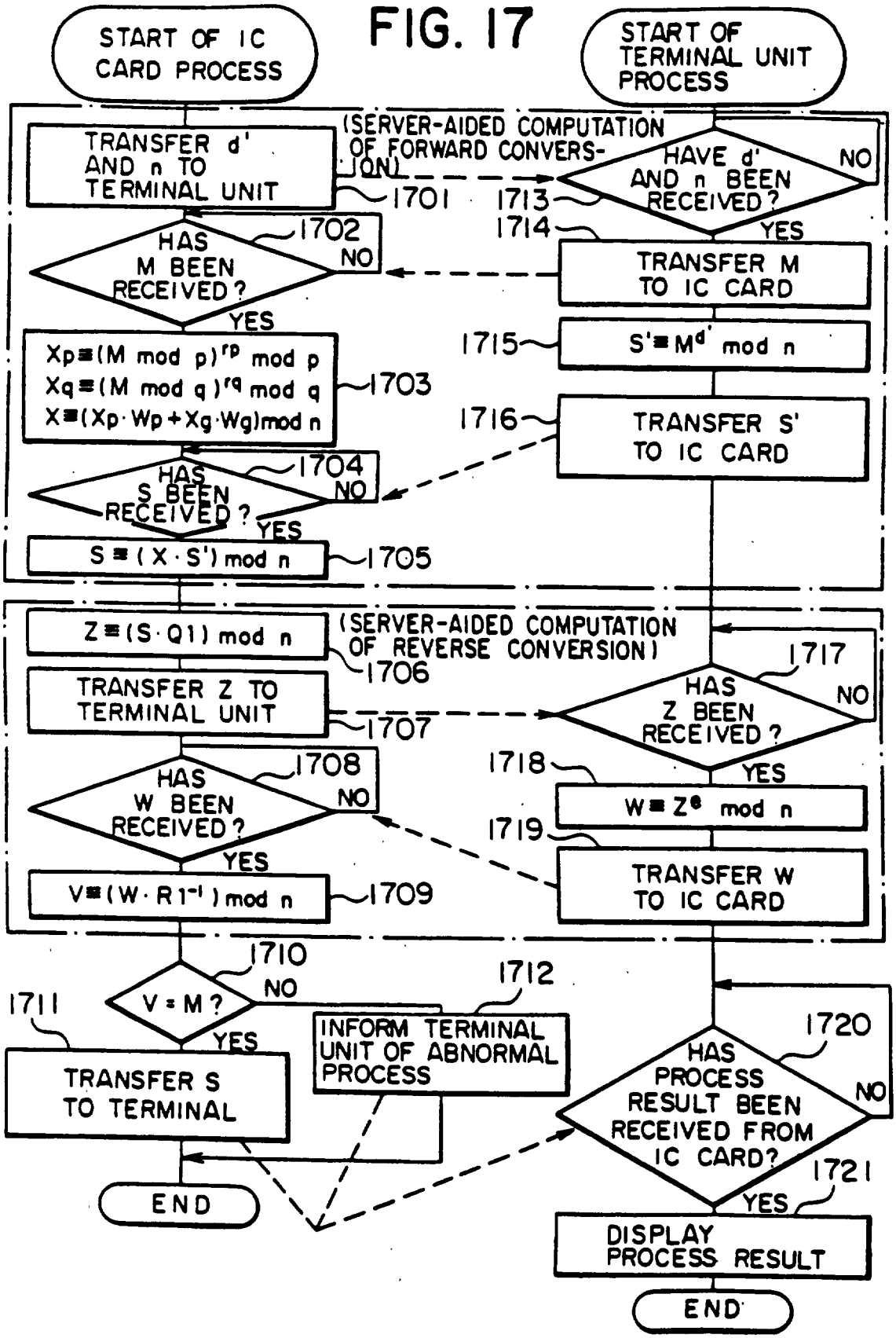

START OF IC CARD PROCESS
START OF TERMINAL UNIT PROCESS
TRANSFER d' AND n TO TERMINAL UNIT
(SERVER-AIDED COMPUTATION OF FORWARD CONVERSION)
1701
HAVE d' AND n BEEN RECEIVED?
1713
NO
YES
1714
TRANSFER M TO IC CARD
1702
HAS M BEEN RECEIVED?
NO
YES
1715
S'≡M^d' mod n
Xp≡(M mod p)^rp mod p
Xq≡(M mod q)^rq mod q
X≡(Xp·Wp+Xg·Wg)mod n
1703
1716
TRANSFER S' TO IC CARD
1704
HAS S BEEN RECEIVED?
NO
YES
S≡(X·S') mod n
1705
Z≡(S·Q1) mod n
(SERVER-AIDED COMPUTATION OF REVERSE CONVERSION)
1706
1717
TRANSFER Z TO TERMINAL UNIT
1707
HAS Z BEEN RECEIVED?
NO
YES
1708
HAS W BEEN RECEIVED?
NO
1718
W≡Z^e mod n
YES
1719
V≡(W·R1^-1) mod n
1709
TRANSFER W TO IC CARD
1710
V=M?
NO
1712
1711
YES
INFORM TERMINAL UNIT OF ABNORMAL PROCESS
1720
TRANSFER S TO TERMINAL
HAS PROCESS RESULT BEEN RECEIVED FROM IC CARD?
NO
1721
YES
DISPLAY PROCESS RESULT
END
END

FIG. 18

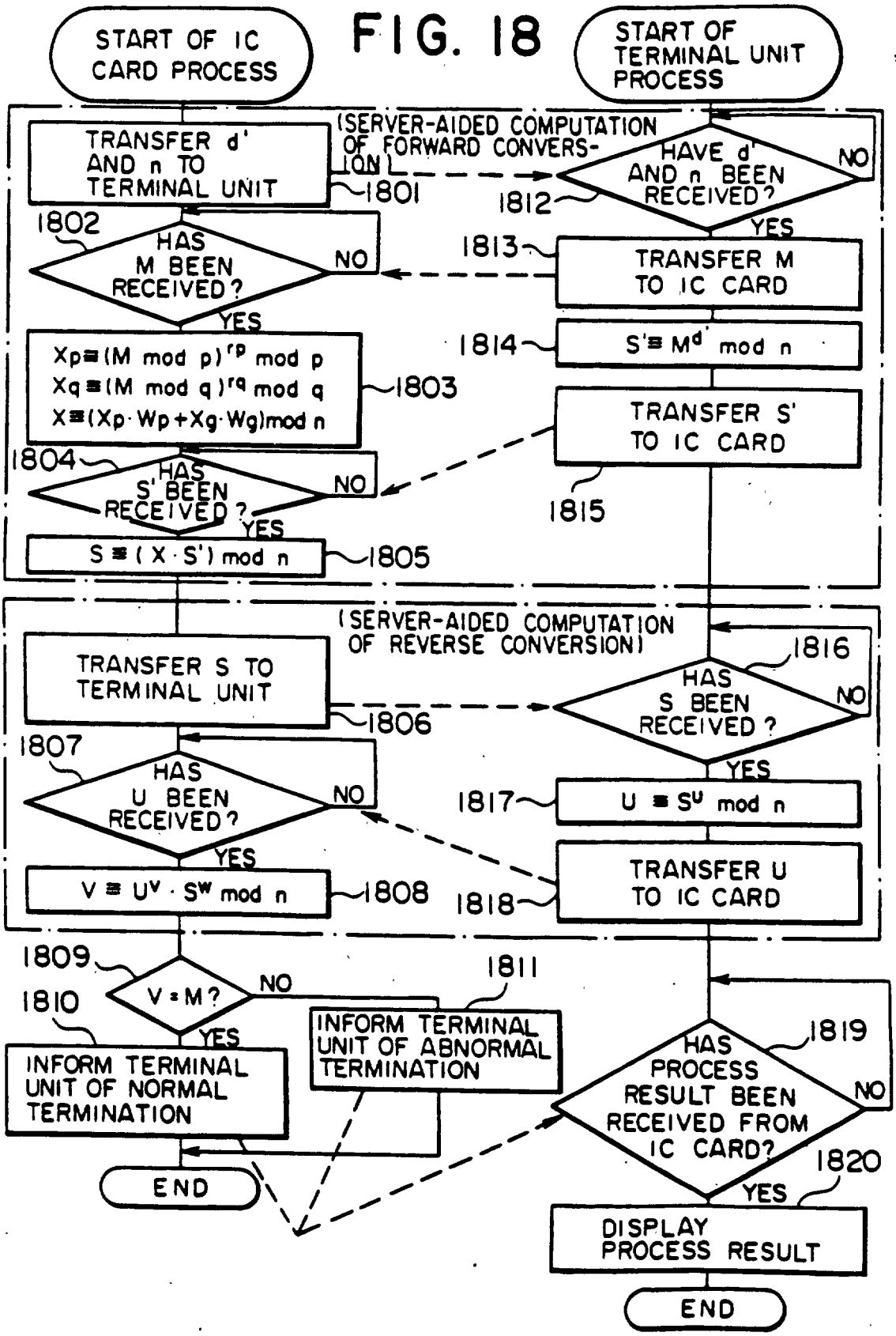

START OF IC CARD PROCESS
START OF TERMINAL UNIT PROCESS
(SERVER-AIDED COMPUTATION OF FORWARD CONVERSION)
TRANSFER d' AND n TO TERMINAL UNIT
1801
HAVE d' AND n BEEN RECEIVED?
1812
NO
YES
1802
HAS M BEEN RECEIVED?
NO
YES
1813
TRANSFER M TO IC CARD
1814
S' ≡ M^d mod n
Xp ≡ (M mod p)^rp mod p
Xq ≡ (M mod q)^rq mod q
X ≡ (Xp·Wp + Xq·Wq) mod n
1803
TRANSFER S' TO IC CARD
1804
HAS S' BEEN RECEIVED?
NO
YES
1815
S ≡ (X·S') mod n
1805
(SERVER-AIDED COMPUTATION OF REVERSE CONVERSION)
TRANSFER S TO TERMINAL UNIT
1806
1816
HAS S BEEN RECEIVED?
NO
YES
1807
HAS U BEEN RECEIVED?
NO
YES
1817
U ≡ S^U mod n
TRANSFER U TO IC CARD
1818
V ≡ U^V · S^W mod n
1808
1809
V = M?
NO
YES
1811
1810
INFORM TERMINAL UNIT OF NORMAL TERMINATION
INFORM TERMINAL UNIT OF ABNORMAL TERMINATION
1819
HAS PROCESS RESULT BEEN RECEIVED FROM IC CARD?
NO
YES
1820
DISPLAY PROCESS RESULT
END
END

FIG. 19

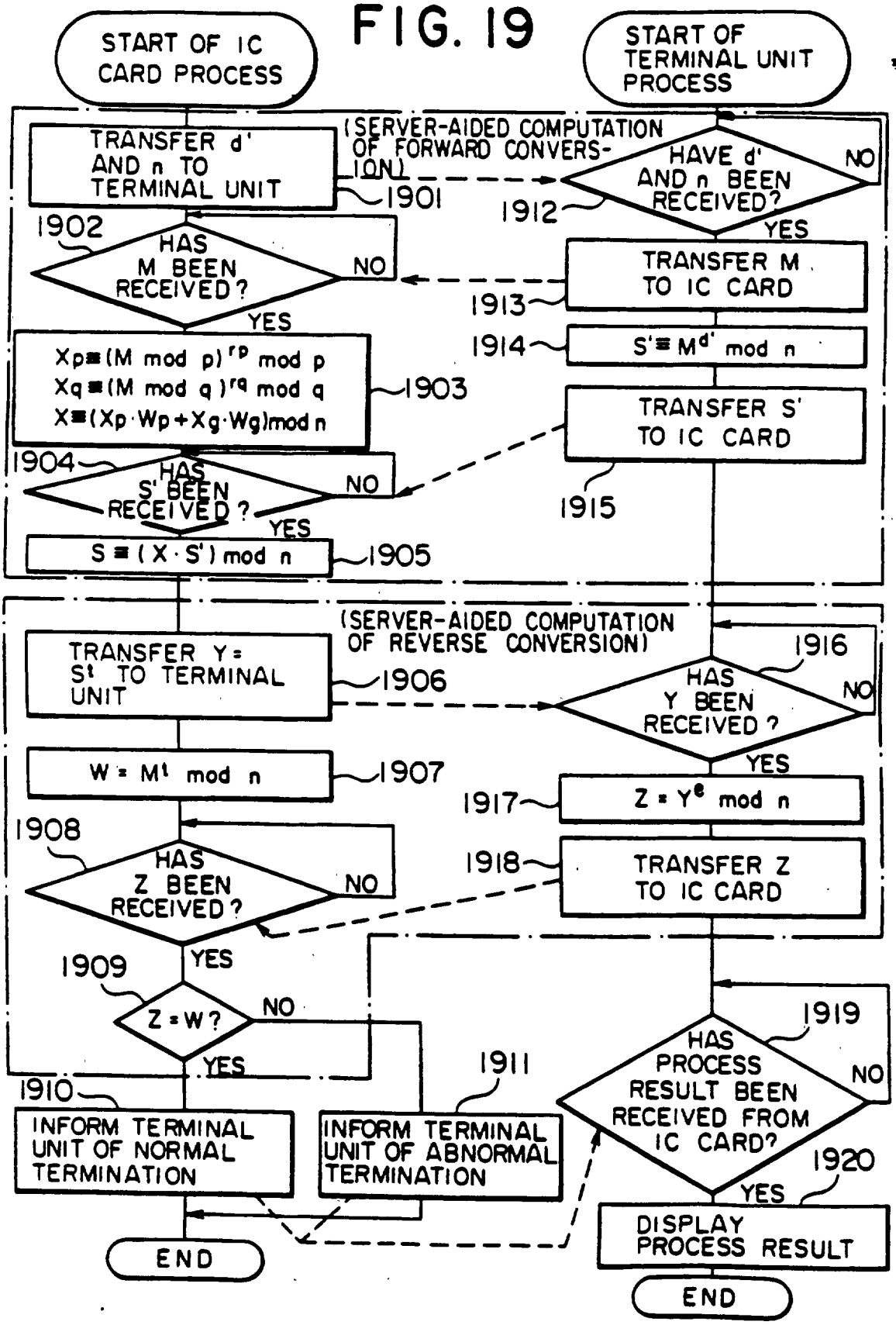

START OF IC CARD PROCESS
START OF TERMINAL UNIT PROCESS
TRANSFER d' AND n TO TERMINAL UNIT
1901
(SERVER-AIDED COMPUTATION OF FORWARD CONVERSION)
HAVE d' AND n BEEN RECEIVED?
1912
NO
YES
1902
HAS M BEEN RECEIVED?
NO
YES
TRANSFER M TO IC CARD
1913
Xp≡(M mod p)^rp mod p
Xq≡(M mod q)^rq mod q
X≡(Xp·Wp+Xg·Wg)mod n
1903
1914
S'≡M^d' mod n
TRANSFER S' TO IC CARD
1904
HAS S' BEEN RECEIVED?
NO
YES
1915
S≡(X·S') mod n
1905
(SERVER-AIDED COMPUTATION OF REVERSE CONVERSION)
TRANSFER Y=S^t TO TERMINAL UNIT
1906
1916
HAS Y BEEN RECEIVED?
NO
YES
W=M^t mod n
1907
1917
Z=Y^e mod n
1908
HAS Z BEEN RECEIVED?
NO
1918
TRANSFER Z TO IC CARD
YES
1909
Z=W?
NO
YES
1919
HAS PROCESS RESULT BEEN RECEIVED FROM IC CARD?
NO
1911
1910
INFORM TERMINAL UNIT OF NORMAL TERMINATION
INFORM TERMINAL UNIT OF ABNORMAL TERMINATION
1920
YES
DISPLAY PROCESS RESULT
END
END

FIG. 20

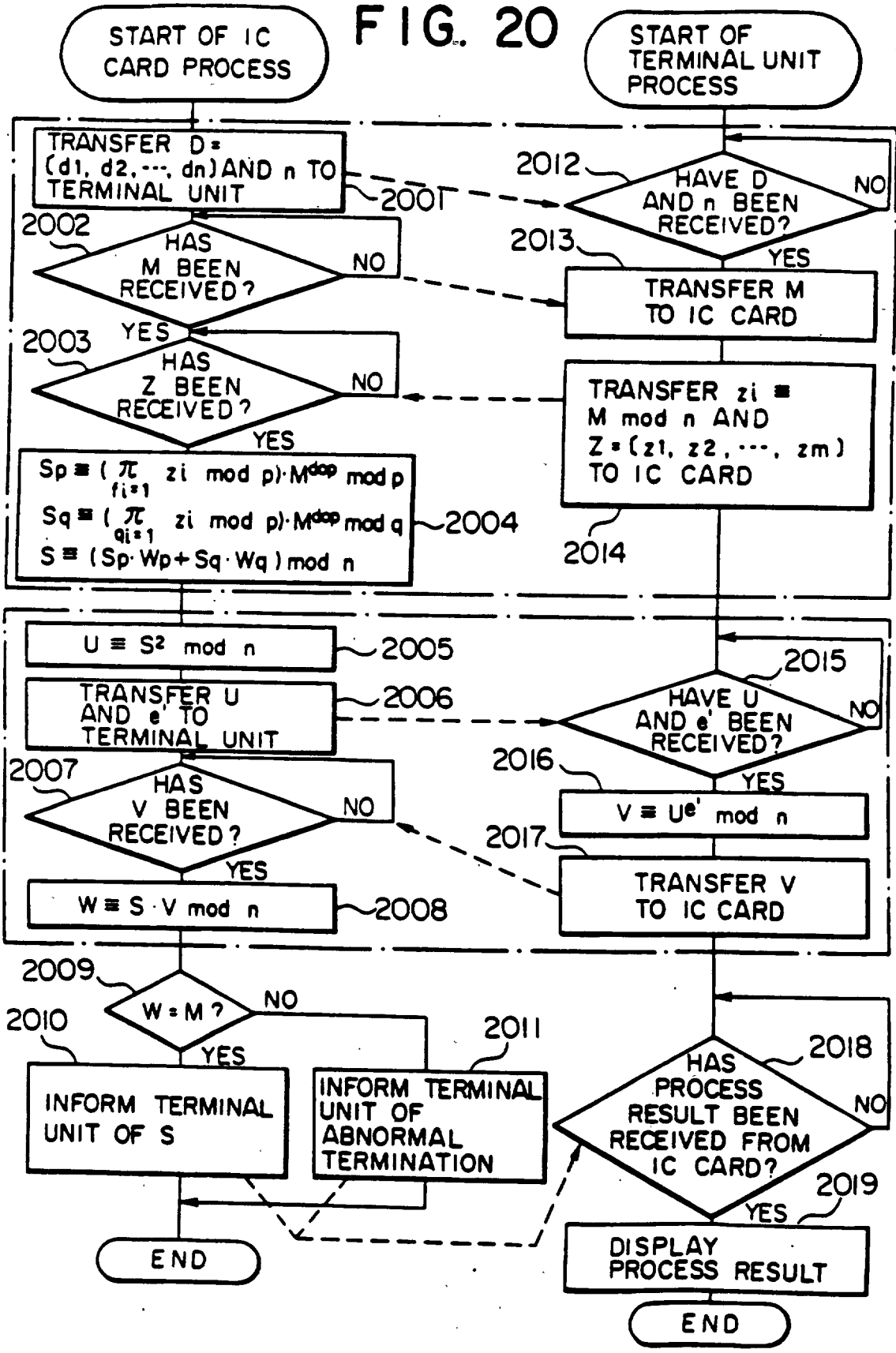

START OF IC CARD PROCESS
START OF TERMINAL UNIT PROCESS
TRANSFER D = (d1, d2, ···, dn) AND n TO TERMINAL UNIT
2001
2002
HAS M BEEN RECEIVED?
NO
YES
2003
HAS Z BEEN RECEIVED?
NO
YES
Sp ≡ (π fi=1 zi mod p)·M^dop mod p
Sq ≡ (π qi=1 zi mod p)·M^dop mod q
S ≡ (Sp·Wp + Sq·Wq) mod n
2004
U ≡ S² mod n
2005
TRANSFER U AND e' TO TERMINAL UNIT
2006
2007
HAS V BEEN RECEIVED?
NO
YES
W ≡ S·V mod n
2008
2009
W = M ?
NO
YES
2010
INFORM TERMINAL UNIT OF S
2011
INFORM TERMINAL UNIT OF ABNORMAL TERMINATION
END
2012
HAVE D AND n BEEN RECEIVED?
NO
YES
2013
TRANSFER M TO IC CARD
TRANSFER zi ≡ M mod n AND Z = (z1, z2, ···, zm) TO IC CARD
2014
2015
HAVE U AND e' BEEN RECEIVED?
NO
YES
2016
V ≡ U^e' mod n
2017
TRANSFER V TO IC CARD
2018
HAS PROCESS RESULT BEEN RECEIVED FROM IC CARD?
NO
YES
2019
DISPLAY PROCESS RESULT
END

FIG. 21

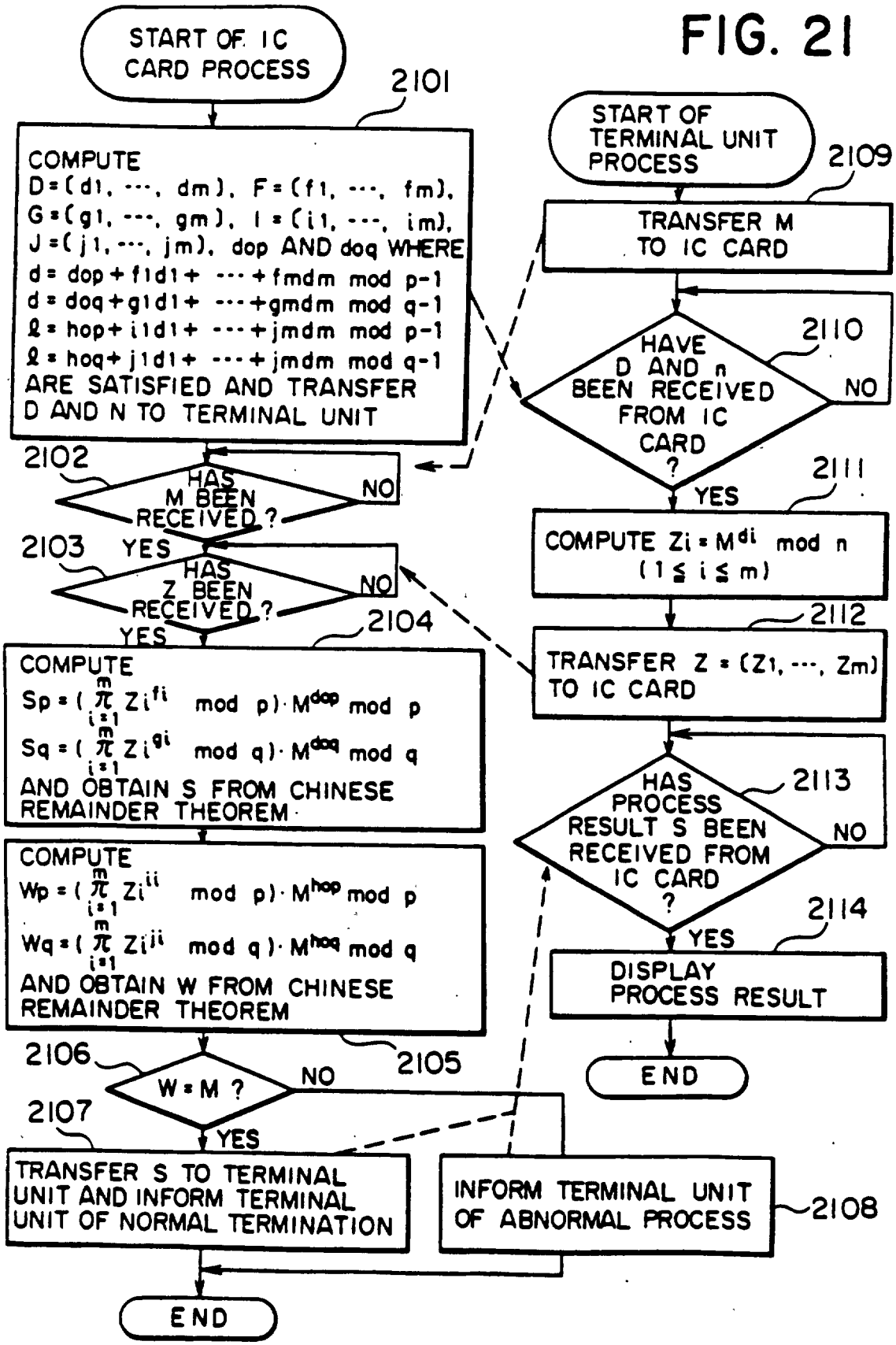
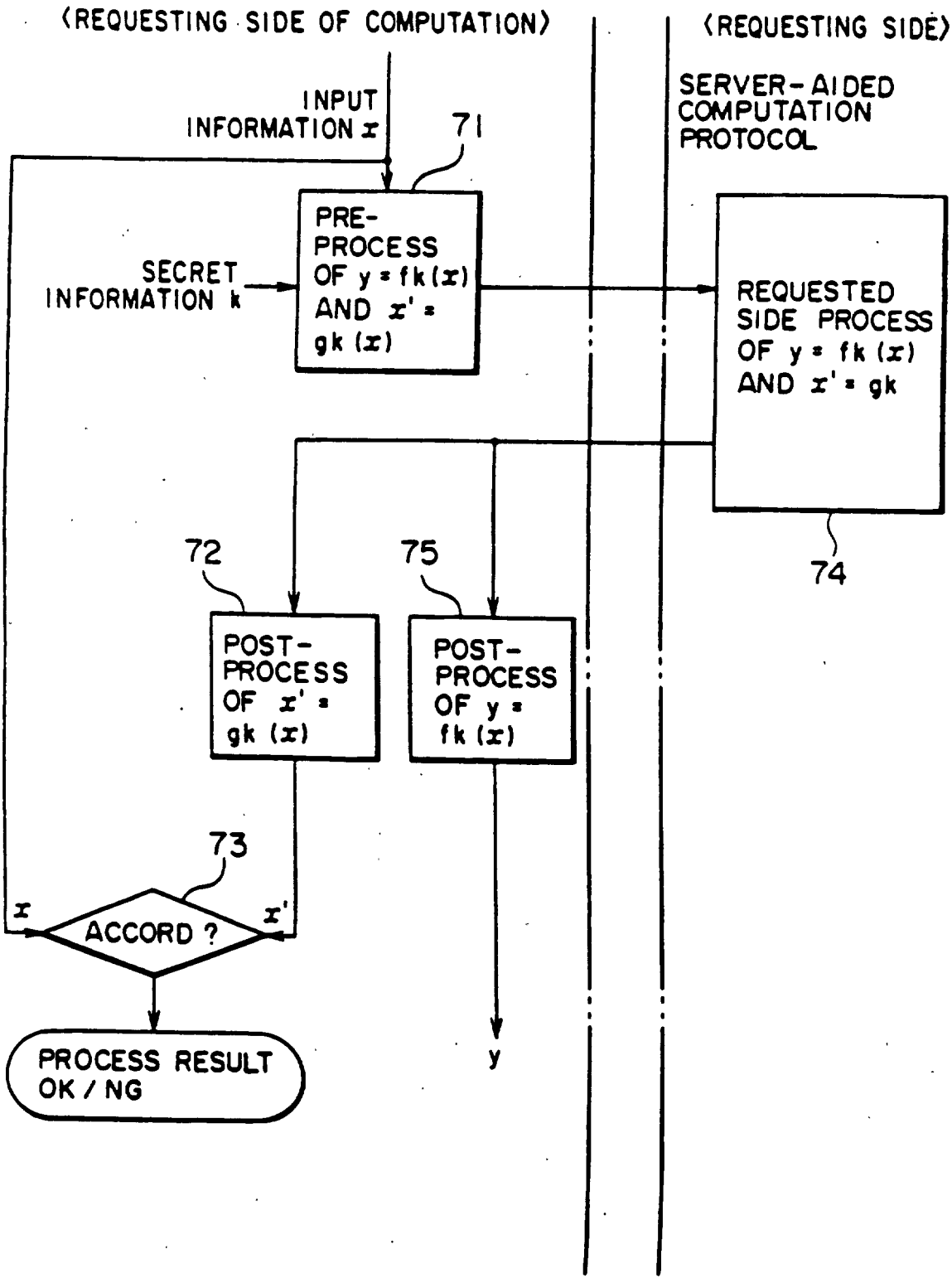

START OF IC CARD PROCESS
2101
COMPUTE D=(d1, ---, dm), F=(f1, ---, fm), G=(g1, ---, gm), I=(i1, ---, im), J=(j1, ---, jm), dop AND doq WHERE d = dop + f1d1 + --- + fmdm mod p-1 d = doq + g1d1 + --- + gmdm mod q-1 ℓ = hop + i1d1 + --- + jmdm mod p-1 ℓ = hoq + j1d1 + --- + jmdm mod q-1 ARE SATISFIED AND TRANSFER D AND N TO TERMINAL UNIT
2102
HAS M BEEN RECEIVED ?
NO
YES
2103
HAS Z BEEN RECEIVED ?
NO
YES
2104
COMPUTE Sp = (∏ Zi^fi mod p)·M^dop mod p
Sq = (∏ Zi^gi mod q)·M^doq mod q
AND OBTAIN S FROM CHINESE REMAINDER THEOREM
COMPUTE Wp = (∏ Zi^ii mod p)·M^hop mod p
Wq = (∏ Zi^ii mod q)·M^hoq mod q
AND OBTAIN W FROM CHINESE REMAINDER THEOREM
2105
2106
W = M ?
NO
YES
2107
TRANSFER S TO TERMINAL UNIT AND INFORM TERMINAL UNIT OF NORMAL TERMINATION
INFORM TERMINAL UNIT OF ABNORMAL PROCESS
2108
END
START OF TERMINAL UNIT PROCESS
2109
TRANSFER M TO IC CARD
2110
HAVE D AND n BEEN RECEIVED FROM IC CARD ?
NO
YES
2111
COMPUTE Zi = M^di mod n (1 ≦ i ≦ m)
2112
TRANSFER Z = (Z1, ---, Zm) TO IC CARD
2113
HAS PROCESS RESULT S BEEN RECEIVED FROM IC CARD ?
NO
YES
2114
DISPLAY PROCESS RESULT
END

FIG. 22


(REQUESTING SIDE OF COMPUTATION)
(REQUESTING SIDE)
SERVER-AIDED COMPUTATION PROTOCOL
INPUT INFORMATION x
71
PRE-PROCESS OF y = fk(x) AND x' = gk(x)
SECRET INFORMATION k
REQUESTED SIDE PROCESS OF y = fk(x) AND x' = gk
74
72
75
POST-PROCESS OF x' = gk(x)
POST-PROCESS OF y = fk(x)
73
x
ACCORD ?
x'
PROCESS RESULT OK / NG
y

SERVER-AIDED COMPUTATION METHOD AND DISTRIBUTED INFORMATION PROCESSING UNIT

FIELD OF THE INVENTION

The present invention relates to a server-aided computation method and a distributed information processing unit for secretly distributing information of a host computer into a plurality of auxiliary units which compute the information.

DESCRIPTION OF THE RELATED ART

When a security service is used with cryptosystem, it is very important to safely distribute and control key information.

The open key cryptosystem RSA proposed by Rivetst et. al has come to public notice as a cryptosystem for solving most of such key distribution problems. The the present invention is based on the RSA cryptosystem, which is described in detail.

KEY GENERATION

First, generate any two large different prime numbers p and q. Generate $n=p\cdot q$ as a product of p and q being generated. Obtain $L=\lambda(n)=LCM\ (p-1, q-1)$ where λ represents Carmichael function and LCM $(p-1, q-1)$ represents the least common multiple of $p-1$ and $q-1$. Select a proper integer e which is relatively prime against L $(3 \leqq e \leqq L-1)$ and obtain the inverse element of multiplication, d, for e in the modulus L.

$$e \cdot d = 1 \bmod L \tag{1}$$

The (e, n) produced in the above method is a key for an encipherment. The key can be deciphered using (d, n).

ENCIPHERMENT AND DECIPHERMENT

A plain text M and a code C are both integers less than n. They are enciphered by the following equation. In the following description, it is necessary to assume that any equal sign represents that a value on the left side is computed by using the right side.

$$C = M^e \bmod n \tag{2}$$

M can be obtained from C in the following equation.

$$M = C^d \bmod n \tag{3}$$

The conversion of the decipherment can be speeded up by using the secret information codes p and q of the receiving side. This method is described in a thesis written by J.J. Quisquater et al, "Fast decipherment algorithm for RSA public-key cryptosystem", Electron. Lett., 18, 21, pp. 905-907 (Oct. 1982).

To compute the value of equation (3), obtain it in moduli p and q rather than directly obtaining it in modulus n. Using the Chinese remainder theorem from the result being obtained, obtain the plain text. (For detail of the Chinese remainder theorem, see a thesis titled "Gendai Angou Riron (Modern Cryptosystem Theory)" by Ikeno, Koyama, et al., The Institute of Electronics, Information and Communication Engmeers, (p. 19).

To practically explain this method, define C_1, C_2, d_1, d_2, m_1, and m_2 as follows

$$C_1 = C \bmod p,\ C_2 = C \bmod q \tag{4}$$

$$d_1 = d \bmod (p-1),\ d_2 = d \bmod (q-1) \tag{5}$$

$$m_1 = M \bmod p,\ m_2 = M \bmod q \tag{6}$$

At the time, the following equations are satisfied.

$$m_1 = C_1^{d_1} \bmod p \tag{7}$$

$$m_2 = C_2^{d_2} \bmod p \tag{8}$$

Thus, the plain text M can be obtained as a root of the following simultaneous congruent expressions.

$$M = m_1 \pmod p \tag{9}$$

$$M = m_2 \pmod q \tag{10}$$

The RSA cryptosystem is also used for a "digital signature." In this case, the equation is expressed as follows,

$$S = M^d \bmod n$$

where a plain text is M and a signature text is S

Although the RSA cryptosystem can be executed in the above method. Now, outline the cryptosystem

A. Open keys e and n which are uniquely assigned to each person are made open to the public in the form of a list. Thus, any one can access the keys.

B. Secret keys d, p, q, and $\lambda(n)$ are kept secret to the public. The person who has the secret keys should take care not to disclose them.

C. Besides the encryption function, a signature function is also provided.

D. To secure the safety of the RSA cryptosystem, it is necessary to select around 100 digits in decimal notation for the number of digits of the secret keys p and q. In this case, n becomes a value of around 200 digits in decimal notation, resulting in requiring a huge processing amount of computation for conversions between the RSA encipherment and decipherment.

As an operation method for maximizing the benefits of the RSA cryptosystem, it is preferable to issue an individual key, to store the key in a portable recording medium, and to have the person who owns the key carry it. In this case, the item B described above is very important in the system operation. As the recording medium of the personal secret key which satisfies the condition of the item B, an IC card is most suitable as a portable and personal computing and recording apparatus. However, when the RSA cryptosystem using IC cards is built up, the following two problems arise.

When a key is stored in the IC card, due to the requirement of the item B above, ideally, it is preferable to execute the RSA decrypting conversion and generate a signature in the IC card. Since the IC card has an access control function which compares a password, when the secret keys d, p, q, and $\lambda(n)$ is converted in the IC card, the secret key d can be prevented from being divulged from the IC card. However, because of the huge processing amount described in the item D and insufficient computation capacity of the IC card, when the RSA code is converted by the IC card, it is difficult to accomplish a practical processing speed. This situation is same even if the high speed method proposed by Quisquater et al. described above. Although it is possible to

consider to mount an RSA dedicated high speed computation LSI on the IC card, an increase of the IC card cost is inevitable.

On the other hand, it is quite easy to use the IC card as a key memory with the access control function. By having a unit with a high computation capacity other than the IC card, for example, a terminal unit execute the complicated code conversion, it is possible to accomplish the practical processing speed. However, in this case, since d is passed to the terminal unit, unless the design, maintenance, and control of the terminal unit are carefully done, d may be divulged to another person via the terminal unit. In addition, d may be unexpectedly stolen via a false terminal unit.

To solve such two problems, recently means for the IC card to efficiently perform the RSA code conversion using only the computation capability of the terminal unit without divulging information relating to the secret key d to the terminal unit have been proposed. This method is named "server-aided computation method" taken from the proposers. Although the server-aided computation method is a wide concept, the method remarkably relating to the RSA code conversion is described in the thesis titled "Anzenna Keisan Iraihou Ni Tsuite (Safety Computation Request Method)", by Kato, Matsumoto, and Imai, Code and Information Security Symposium Material F-3, February 1988. The method is described in the following.

As a preparation, firstly obtain r_p, r_q and R which satisfy the following equations.

$$r_p = R^{-1} \bmod (p-1) \tag{11}$$

$$r_q = R^{-1} \bmod (q-1) \tag{12}$$

However, when it is defined that $\chi(r) = l(r) + w(r) - 2$, r_p and r_q are selected so that

$$\chi(r_p) = \chi(r_q) \tag{13}$$

becomes a small value; l (r) represents the bit length of r; w (r) represents the hamming weight of r; and $\chi(r)$ represents the number of times of the modulo- multiplications necessary for the modulo- exponentiation where r is an exponent.

In addition, compute the following equations.

$$W_p = q\,(q^{-1} \bmod p) \bmod n,$$

$$W_q = p\,(p^{-1} \bmod q) \bmod n \tag{14}$$

In the IC card, r_p, r_q, R, d, p, q, $\lambda(n)$, n, W_p, and W_q have been stored.

Then, using the following equation instead of d

$$d' = d \cdot R \bmod \lambda(n) \tag{15}$$

the IC card requests the terminal unit to compute M' where C is converted by d' where

$$M' = C^{d'} \bmod n. \tag{16}$$

The terminal unit returns M' being computed to the IC card. The IC card converts M' into the plain text M by using the following equation.

$$M = \{(M' \bmod p)^{r_p} \bmod p\}\, W_p + \{(M' \bmod q)^{r_q} \bmod q\}\, W_q) \bmod n \tag{17}$$

Since r_p and r_q have been selected so that the value of the equation (13) becomes small, the modulo- exponentiation computation can be effectively performed by the IC card whose computation capacity is relatively small. In addition, to the terminal unit, the IC card sends d' converted by the equation (15) rather than d, thereby enhancing the degree of safety. When the server-aided computation is performed in the manner described above, the code conversion can be effectively performed with the computation capacity of the terminal unit as well as increasing the degree of safety of the secret key d.

As described above, when secret information such as the RSA cryptosystem is computed, if a unit computes a huge processing amount of information, it takes much computation time. For example, if an IC card whose computation capacity is relatively small executes such information, it takes much computation time. In addition, it is possible to consider to use an auxiliary unit as well as the main computation unit to share the computation load of the secret information between them so as to reduce the computation time. However, if the secret information is directly sent to the auxiliary unit, it may be stolen by the unit or a third party. For example, when d is sent to an external unit which can execute the RSA computation at a high speed, the secret information necessary for the decryption and generation of a digital signature is known by the external unit and thereby the information may be invalidly used.

On the other hand, the secret information using the "server-aided computation" which has been proposed can be effectively converted using the computation capacity of an external unit without divulging the secret information. However, the external unit is not always reliable and the communication information may be changed by a third party. Thus, the requesting side cannot detect an invalidity of the requested side and a change of communication information by the third party. Consequently, the validity of the server-aided computation becomes doubtful.

SUMMARY OF THE INVENTION

An object of the present invention is to provide a server-aided computation method and a distributed information processing unit for preventing secret information from divulging to a requested side of the computation, for effectively computing the secret information using the computing capacity of the requested side, and for really validating the server- aided computation.

For example, the RSA cryptosystem can be used both for enciphering messages and for generating digital signatures.

The first invention is a server-aided computation method using a main unit for processing secret information and at least one auxiliary unit for supporting a computation that said main unit executes, said method comprising the steps of;

generating d' from a secret key d' using m random numbers R_i (where $i = 1, \ldots, m$) generated by said main unit having secret keys n and d;

transferring d' and n from said main unit to said auxiliary unit;

computing the following equation from a message block C in said auxiliary unit

$$M' = C^{d'} \bmod n$$

computing X using said random numbers R_i and n in said main unit while computing M' in said auxiliary unit;

transferring M' from said auxiliary unit to said main unit; and

computing a message block M using the following equation in said main unit

$$M = M' \cdot X \bmod n$$

The second invention is a server-aided computation method using a main unit for processing secret information and at least one auxiliary unit for supporting a computation that said main unit executes, said method comprising the steps of:

generating d' from a secret key d using m random numbers R_i (where $i = 1, \ldots, m$) generated by said main unit having secret keys n and d;

transferring d' and n from said main unit to said auxiliary unit;

computing the following equation from a message block C in said auxiliary unit

$$M' = C^{d'} \bmod n$$

computing X^{-1} using said random numbers R_i and n in said main unit while computing M' in said auxiliary unit;

transferring M' from said auxiliary unit to said main unit; and

computing a message block M using the following equation in said main unit

$$M = M' \cdot X^{-1} \bmod n$$

The third invention is a server-aided computation method using a main unit for processing secret information and at least one auxiliary unit for supporting a computation that said main unit executes, said method comprising the steps of:

generating d' from a secret key d' using m random numbers R_i (where $i = 1, \ldots, m$) generated by said main unit having secret keys n and d;

transferring d' and n from said main unit to said auxiliary unit;

computing the following equation from a message block C in said auxiliary unit

$$M' = C^{d'} \bmod n$$

computing X and X^{-1} using said random numbers R_i and n in said main unit while computing M' in said auxiliary unit;

transferring M' from said auxiliary unit to said main unit; and

computing a message block M using M', X, and X^{-1}.

In the above server-aided computation methods, the auxiliary unit only knows d' and n which have been open and C and M'. Since the auxiliary unit cannot directly know the secret key d, it is impossible to devise a function which causes the auxiliary unit to steal the secret key d. The section of the computations for obtaining X_{pi}, X_{qi}, or X_i that the main unit executes can be conducted independently from the computations that the second unit executes. In addition, by restricting the bit length of random numbers r_{pi} and r_{qi}, the amount of the computation for obtaining X_{pi} and X_{qi} can be reduced.

The fourth invention is a server-aided computation method using a main unit for processing secret information and at least one auxiliary unit for supporting a computation that said main unit executes wherein $S = M^d \bmod n$ where a positive integer M less than an integer n given by a secret positive integer d which is sent to said auxiliary unit is raised to n-th power in accordance with an algebraic system where the given positive prime number or a composite number n is a modulus, said method comprising the steps of:

separating said integer n into k ($k \geq 1$) positive integers n_j (where $j = 1, \ldots, k$) each of which is relatively prime;

(b) separating said positive integer d into $(m-1) \times k$ non-negative integers $D_{ij} = \{d_{j0}, f_{j1}, F_{j2}, \ldots, f_{jm}\}$ which are secret information only for said main unit and which satisfy the following k sets of equations

$$d = d_{j0} - f_{j1} \cdot d_{j1} - f_{j2} \cdot d_{j2} - \ldots - f_{jm} \cdot d_{jm} \pmod{\lambda(n_j)}$$

where $j = 1, \ldots, k$ and $\lambda(n_j)$ is the Carmichael function of said positive integer n) and sets of the following $m \times k$ positive integers which are transferred to said auxiliary unit

$$D_{2j} = \{d_{j1}, d_{j2}, \ldots, d_{jm}\} (j = 1, \ldots, k)$$

(c) computing $Y_{ji} = M^{d_{ji}} \bmod n$ where $i = 1, \ldots, m$ and $j = 1, \ldots, k$ in said auxiliary unit and sending the results to said main unit; and

(d) computing in said main unit the following equation using k values $Y_{j0} = M^{d_{j0}} \bmod n$ and above Y_{ji} which have been computed by said main unit

$$S = Y_{j0} \cdot Y_{j1}^{f_{j1}} \cdot Y_{j2}^{f_{j2}} \cdots Y_{jm}^{f_{jm}} \bmod n_j$$

where $j = 1, \ldots k$ and obtaining a result S which satisfies the k simultaneous equations relating to S.

The fourth invention can be also applied when said integer n is a prime number.

The fourth invention can be applied when said integer is a product of two prime numbers.

In the fourth invention, since D_{ij} is kept secret to the auxiliary unit which supports the main unit, the auxiliary unit cannot know D_{ij} unless it tries to execute the round robin method. Thus, it is possible for the main unit to execute the computation without divulging the secret information to the auxiliary unit.

In addition, when the computation speed of the auxiliary unit is satisfactorily high, the required computation can be executed at a higher speed than that executed only by the main unit.

As a special case of the fourth invention, when $m \times n$ non-negative integers $f_{j1}, f_{j2}, \ldots, f_{jm}$ are selected to 1 or 0, the computation load of the main unit can be reduced.

If a condition where the value d_{ij} is defined so that $d_{ij} = d_{uv}$ is imposed for at least one set of integer pairs (i, j) and (u, v) (where $1 \leq i, u \leq k$, $1 \leq j$, and $v \leq m$), when the auxiliary unit executes $M^{d_{ij}} \bmod n$, it does not need to execute $M^{d_{uv}} \bmod n$ and thereby the entire computation load can be reduced.

The fifth invention is a distributed information processing unit having a main unit for processing secret information and at least one auxiliary unit for supporting a computation that said main unit executes, for executing a distributed process without divulging conversion of said secret information other than said main unit, said distributed information processing unit comprising:

conversion means for executing said conversion of input information and reverse conversion means for reversely converting the conversion results; and

verification means for comparing the reverse conversion results of said reverse conversion means with said input information so as to verify the conversion results of said conversion means.

The sixth invention is a distributed information processing unit having a main unit for processing secret information and at least one auxiliary unit for supporting a computation that said main unit executes, for executing a distributed process without divulging conversion of said secret information other than said main unit, said distributed information processing unit comprising:

a plurality of conversion means for executing said conversion of input information; and

verification means for mutually comparing the conversion results of said plurality of conversion means.

In the fifth and sixth inventions, by comparing the result of the reverse conversion obtained by the reverse conversion means with the input information or by mutually comparing a plurality of conversion results, the main unit can effectively execute the conversion with a support of the computation capacity of the requested side without divulging the secret information to the auxiliary unit of the requested side and determine whether the computation results are correct or not.

The seventh invention is a distributed information processing unit having a main unit for processing secret information and at least one auxiliary unit for supporting a computation that said main unit executes, for executing a distributed process without divulging conversion of said secret information other than said main unit, said distributed information processing unit comprising:

first conversion means for executing said conversion of input information;

second conversion means for executing an identity conversion;

and comparison means for comparing the conversion results of said second conversion means with said input information so as to verify the conversion results of said first conversion means.

In the seventh invention, by comparing the result obtained by conversion means using vectors created according to a special rule with input information, the main unit can execute the conversion without divulging the secret information intrinsic to the main unit, effectively execute the computation with an assistance of the computation capacity of the auxiliary unit of the requested side, and easily determine whether the computation request is correct or not without a communication for the verification and an assistance of the requested side.

BRIEF DESCRIPTION OF DRAWINGS

FIG. 1 is a processing flow chart of a server- aided computation method embodying the first invention;

FIG. 2 is a perspective view of a terminal unit 2;

FIG. 3 is a block diagram showing the structure of an IC card 9;

FIG. 4 is a block diagram showing the structure of the terminal unit 2;

FIG. 5 is a flow chart showing the process of the terminal unit 2;

FIG. 6 is a process flow chart of a server- aided computation method embodying the second invention.

FIG. 7 is a block diagram showing a computation section and the unit;

FIGS. 8 and 9 are diagrams showing a process where the fourth invention is structured by the IC card and the terminal;

FIG. 10 is a chart showing a process time characteristic;

FIG. 11 is a block diagram showing an outline of the fifth to seventh inventions;

FIGS. 12 and 13 are a process flow chart embodying the fifth invention and an outline diagram showing the general structure.

FIGS. 14 and 15 are a process flow chart and an outlined diagram of the general structural example of another embodiment;

FIGS. 16, 17, and 18 are an outlined diagram of the general structural example, a flow chart of an example of the process, and a flow chart of another example of the process of another embodiment, respectively.

FIGS. 19 and 20 are flow charts showing another embodiment; and

FIGS. 21 and 22 are a flow chart of the process and an outlined diagram of the general structural example of another embodiment, respectively.

DETAILED DESCRIPTION OF PREFERRED EMBODIMENT

An embodiment of the present invention is described in the following. For convenience of the description, as shown in FIG. 1, it is assumed that the requesting side of computation is the IC card and the requested side of computation is the terminal. However, the requesting side and requested side can be freely structured using devices and software in the range of the present invention. The embodiment is described as a server-aided computation where a cipher text is deciphered to a plain text. However, the present invention is also applicable to generate a digital signature by using the same conversion.

FIG. 2 is a perspective view of a terminal unit 2.

As shown in the figure, the terminal unit 2 is composed of a main unit 1, a display 3, a keyboard 5, and a reader/writer 7. An IC card 9 is inserted into the reader writer 7. A floppy disk 11 is inserted into the main unit 1.

FIG. 3 is a block diagram showing the structure of the IC card 9. The IC card 9 is provided with an I/O contact 13, a CPU 15, a data memory 17, and a program memory 19.

FIG. 4 is a block diagram showing the structure of the main unit 1. The main unit 1 is composed of a display controller 21, a central processing unit 23, a main memory 25, a first communication port 27, a second communication port 29, a floppy disk driver 31, and a keyboard (I/O) 33, each of which is connected via an internal data bus 35. The display controller 21 controls the display 3. The central processing unit 23 controls the entire terminal unit 2. The main memory 25 stores programs that the central processing unit 23 executes and data used for the programs.

The first communication port 27 is connected to a communication line 12. The communication line 12 is connected to another terminal unit 2. The second communication port 29 is connected to the reader/writer 7. The floppy disk driver 31 drives the floppy disk 11. The keyboard (I/O) 33 is connected to the keyboard 5.

Next, using the terminal unit 2 and the IC card 9, the server-aided computation method is described.

As a preparation, a proper set of random numbers, r_p, r_q, and R, which satisfy the equations (18) and (19) similar to the equations (11) and (12) is obtained.

$$r_p = R \bmod (p-1) \quad (18)$$

$$r_q = R \bmod (q-1) \quad (19)$$

At the time, as described above, the restriction where the result of the following equation is relatively small is applied.

$$\chi(r_p) = \chi(r_q) \quad (20)$$

The simultaneous equations (18) and (19) are solved after r_p and r_q are properly defined. The existing condition of solution and the solution are described on pages 31-35 of Sadaharu Takagi's, "SHOTO SEISUURON KOUGI (Elementary Theory of Numbers)", Kyoritsu Syuppan. The solution of the simultaneous equations is uniquely obtained assuming that $L = LCM(p-1, q-1)$ is a modulus. If necessary, the following equations are also computed.

$$w_p = q\,(q^{-1} \bmod p) \bmod n,$$

$$w_q = p\,(p^{-1} \bmod q) \bmod n \quad (21)$$

As described later, w_p and w_q are constants which should not be always prepared. In the present embodiment, r_p, r_q, d, p, q, $\lambda(n)$, n, d', w_p, and w_q have been stored in the IC card.

As a simplest and most effective embodiment, the conversion where d' is obtained from d is defined using the following equation.

$$d' = (d-R) \bmod \chi(n) \quad (22)$$

FIG. 5 is a process flow chart showing a process of the terminal unit 2.

First, the user faces the terminal unit 2 and then inserts the own IC card 9 into the reader writer 7 for the IC card 9 which is connected to the terminal unit 2 (in step 501).

The user presses proper keys on the terminal unit 2 to inform the terminal unit 2 that the operation thereof is started. At the time, a clock and power are supplied to the IC card 9 via the reader writer 7. After the IC card 9 is initialized (in step 502), the IC card 9 enters a communication waiting state. The terminal unit 2 requires the user to enter the own password to verify whether the the user of the IC card 9 is valid or invalid (in step 503).

When the password is not entered (in step 504), the elapsed time is checked (in step 505). When the specified time elapsed, a timeout occurs. Otherwise, the control returns back to step 503. When the password is entered, the password is transferred to the IC card 9 (in step 506). The password is compared with the registered password stored in the IC card 9 and the compared result is transferred to the terminal unit 2 (in step 507). When the compared result is OK (in step 508), the IC card 9 becomes a valid state. When the compared result is not OK, the IC card 9 becomes an invalid state. When the user enters a command (in step 509), whether the entered command is an end command or not (in step 510) is checked and a command subroutine is executed (in step 511).

An execution of the command subroutine is described in the following.

The process that follows is shown in FIG. 1.

The terminal unit 2 transfers the cipher text C to the IC card 9 (in step 101) and reads d' and n which have been written in the memory of the IC card 9 (in step 102). The terminal unit 2 computes M' from the cipher text C using the two pieces of information (in step 103).

$$M' = C^{d'} \bmod n \quad (23)$$

The terminal unit 2 sends M' which has been computed to the IC card 9 (in step 105).

On the other hand, the IC card 9 obtains a constant X according to the equations (24) to (26) along with the computation in the terminal unit 2 (in step 104).

$$X_p = (C \bmod p)^{r_p} \bmod p \quad (24)$$

$$X_q = (C \bmod q)^{r_q} \bmod q \quad (25)$$

The above equations are computed and X is obtained using the following equation.

$$X = \{((X_p)^{r_p} \bmod p)\, w_p + ((X_q)^{r_q} \bmod q)\, w_q\} \bmod n \quad (26)$$

The equations (24) to (26) are supplementarily described in the following.

Although it can be considered that the equations (24) and (25) are simultaneous equations for X, X which satisfies the above two equations can be uniquely obtained from the Chinese remainder theorem. The one solution of the equations is the right side of the equation (26). X has been obtained using the two auxiliary variables w_p and w_q.

However, the method for obtaining X which satisfies the equations (24) and (25) is not limited to the above method. For example, another solution is represented in the theses on pages 905-907 of J. J. Quisquater et al., "Fast decipherment algorithm for RSA public-key cryptosystem", Electron. Lett., 18, 21, October 1982.

Therefore, the obtainment of X and use of the auxiliary variables w_p and w_q by the equation (26) are not essential. Rather, the obtainment of X which satisfies both the equations (24) and (25) is essential. Thus, this embodiment does not limit the method for obtaining X when the first invention is actually accomplished.

Although the plain text M that the IC card 9 needed to obtain can be obtained by converting the cipher text C using the following equation.

$$M = C^{d} \bmod n \quad (27)$$

where M is obtained from M' computed by the terminal unit 2 and X computed by the IC card 9 using the following equation (in step 106).

$$M = (M' \cdot X) \bmod n \quad (28)$$

In this example, the computation is executed in the IC card 9. The IC card 9 transfers M, which has been obtained, as the deciphered result to the terminal unit 2 (in step 107).

The terminal unit 2 displays the deciphered result on the display and writes it to the auxiliary storage unit to complete the decipher process sequence. The user removes the IC card 9 from the reader writer 7 and completes the operation.

In the present embodiment, since the terminal unit 2 can easily obtain X from M and M' in the process, it is necessary to note that the computation of the equation (28) can be executed in the terminal unit 2 rather than the IC card 9. In this process, the terminal unit 2 does not transfer M' to the IC card 9. Rather, the IC card 9 transfers X to the terminal unit 2.

In the above process, it is obvious that M can be correctly computed using the equation (28) as described below.

From the equations (24) to (26), the following equation is satisfied by the Chinese remainder theorem which has been mentioned as [Related Art].

$$X = C^R \bmod n \tag{29}$$

On the other hand, the following equation is also satisfied.

$$M = C^d \bmod n = C^{d-R \bmod \lambda(n)} \bmod n = C^{d-R} \bmod n = C^d C^{-R} \bmod n \tag{30}$$

From the equations (29) and (30), the following equation is also satisfied.

$$(M \cdot X) \bmod n = (C^d C^{-R} \cdot C^R) \bmod n = C^d \bmod n = M \tag{31}$$

Thus, it is obvious that the equation (28) is satisfied.

Then, the computation load is considered in the following. The computations from the equations (18) to (22) can be prepared before the conversion is started. After the cipher text is given, it is possible to consider the computations only for portions which are executable. As steps to be executed after C is given, (1) Obtainment of X by the IC card 9 using the equations (24) to (26); (2) Computation of the equation (23) by the terminal unit 2; and (3) Computation of the equation (28) by the The IC card 9. In the above three steps, the computation (2) that the terminal unit 2 performs requires the largest computation load.

Practically, this value can be represented as $\chi(d')$ However, when n is 512 bits, a modulo-multiplication for 512 bits should be executed 1024 times in the worst case. The step that requires the next largest computation load is (1). The equations which require major computation load in step (1) are the equations (24) and (25). They require a modulo-multiplication for 256 bits $\chi(r_p) + \chi(r_q)$ times. By selecting small values for r_p and r_q in advance, the computation load can be reduced. The computation in step (3) that the IC card 9 performs is a modulo-multiplication for 512 bits one time. The major portions of the computation load are the computations in steps (1) and (2). In the first invention, particularly note that the step (1) and the step (2) are independent and they can be executed in parallel. For example, when a general purpose personal computer takes 30 seconds for executing the computation in step (2), if the bit length of r_p and r_q is determined to a proper value and thereby the computation time of step (1) executed by the IC card becomes approx. 30 seconds, the total computation time for the decipherment could become around 30 seconds. When the periods of time necessary for the computations in steps (1) to (3) are represented as T1, T2, and T3, respectively, the total computation time T can be generally represented as the following equation.

$$T = Max(T1, T2) + T3 \simeq Max(T1, T2) \tag{32}$$

where Max (A, B) is a function which selects a larger one of A and B.

In the first embodiment, the process for obtaining M using the set of M' and (X_p, X_q) or using values equivalent to the set is not limited to the method described above. Another method for obtaining M is described in the following.

Using the following two equations, from M,

$$M'_p = M \bmod p \tag{33}$$

$$M'_q = M \bmod q \tag{34}$$

by computing M'_p and M'_q, the following two equations are obtained.

$$M_p = M'_p X_p \bmod p \tag{35}$$

$$M_q = M'_q X_q \bmod p \tag{36}$$

By simultaneously solving the equations (35) and (36), the required M can be obtained.

Then, as an embodiment of the second invention, according to FIG. 6, the conversion where d' is obtained from d is defined using the following equation.

$$d' = (d - R) \bmod \lambda(n) \tag{37}$$

In the description that follows, the startup of the terminal unit 2 and the initialization of the IC card 9 are omitted. Rather, only the process for the computation is sequentially described. It is assumed that R used in the first embodiment is the same as that in this embodiment Like the embodiment of the first invention, the terminal unit 2 transfers the cipher text C which has been input from the outside to the IC card (in step 601). The terminal unit 2 receives d' and n from the IC card 9 (in step 602). Like the first embodiment, the terminal unit 2 computes M' which is given in the following equation (in step 603).

$$M' = C^{d'} \bmod n \tag{38}$$

The terminal unit 2 sends M, computed therein back to the IC card 9 (in step 607).

On the other hand, the IC card 9 computes the following equations (in step 604)

$$X_p = (C \bmod p)^{r_p} \bmod p \tag{39}$$

$$X_q = (C \bmod q)^{r_q} \bmod p \tag{40}$$

and obtain X using the following equation (in step 605)

$$X = (((X_p)^{r_p} \bmod p) \cdot w_p + ((X_q)^{r_q} \bmod q) \cdot w_q) \bmod n \tag{41}$$

In addition, by solving the following equation

$$X^{-1} \cdot X = 1 \bmod n \ (\text{in } 42)$$

X^{-1} is obtained (in step 606). This solution is named the extended Euclidean algorithm. For details, see the thesis "Gendai Angou Riron" described above.

The value to be obtained by the IC card 9, namely M, is expressed as follows.

$$M = C^d \bmod n \tag{43}$$

This value is obtained from the following equation using M' which has been computed by the terminal unit 2 and X^{-1} which has been computed by the IC card 9 (in step 608).

$$M=(m\cdot X^{-1}) \bmod n \quad (44)$$

The IC card 9 transfers the result being obtained to the terminal unit 2 and completes the process (in step 609).

In the following, it will become obvious that M can be correctly computed from the equation (44).

From the equations (35) to (38) and the Chinese remainder theorem, the following equation is satisfied.

$$X^{-1}=C^{-R} \bmod n \quad (45)$$

On the other hand, since M' can be expressed as follows,

$$\begin{aligned} M' &= C^d \bmod n \quad (46)\\ &= C^{d-R \bmod \lambda(n)} \bmod n\\ &= C^{d-R} \bmod n\\ &= C^d C^R \bmod n \end{aligned}$$

From the equations (29) and (30),

$$\begin{aligned} (M'\cdot X^{-1}) \bmod n &= (C^d C^R\ C^{-R}) \bmod n \quad (47)\\ &= C^d \bmod n\\ &= M \end{aligned}$$

Thus, it is obvious that the equation (28) is satisfied.

In the embodiment of the second invention, the process for obtaining M using M' and the set of (X_p, X_q) or a value equivalent to this set is not limited to the method described above.

For example, by computing C^{-1} using the extended Euclidean algorithm in advance, the following equations can be computed.

$$X_p^{-1}=(C^{-1} \bmod p)^{r_p} \bmod p \quad (48)$$

$$X_q^{-1}=(C^{-1} \bmod q)^{r_q} \bmod q \quad (49)$$

Using the results, X^{-1} can be computed. M can be obtained in the same manner as the equation (40).

In addition, from the following two equations using M

$$M_p=M' \bmod p \quad (50)$$

$$M_q=M' \bmod q \quad (51)$$

M'_p and M'_q are obtained and thereby the following equations are satisfied.

$$M_p=M'_p X_p^{-1} \bmod p \quad (52)$$

$$M_q=M'_q X_q^{-1} \bmod q \quad (53)$$

By simultaneously solving the equations (48) and (49), the required M can be obtained. The effect of the second invention is the same as that of the first invention.

As an embodiment of the third invention, a more generalized method is described. In the first and the second embodiments, the unique random number R has been used in the algebraic system where the Carmichael function (n) is a modulus. In the embodiment of the third invention, a general format using m random numbers R_i ($i=1, \ldots, m$; $m \geq 1$) is described. Firstly, it is assumed that each random number R_i satisfies the following equations.

$$R_i \bmod p = r_{ip} \quad (54)$$

$$R_i \bmod q = r_{iq} \quad (55)$$

Like the embodiment of the first invention, it is also assumed that the value of the following expression is properly restricted.

$$\sum_{i=1}^{m} \{\chi(r_{ip}) - \chi(r_{iq})\} \quad (56)$$

As suggested in the equations (54) to 56), it is necessary to define r_{ip} and r_{iq} and then obtain R_i.

Using R_i obtained in the above manner, each conversion f_i from x to y is defined.

$$y=f_i(R_i, x) \quad (57)$$

Using the resultant conversion where m conversions are composed, d, is converted into d'.

$$d'=f_m(R_m, \ldots f_2(R_2, f_1(R_1, d)) \ldots) \quad (58)$$

As the practical definition of f_i, the following three types can be used.

$$y=x(R_i^{-1}) \bmod \lambda(n) \quad (59)$$

$$y=(x-R_i) \bmod \lambda(n) \quad (60)$$

$$y=(x-R_i) \bmod \lambda(n) \quad (61)$$

The equation (59) is the function which has been described. The equations (60) and (61) are the functions which have been represented in the embodiments of the first and the second inventions. By using any combination of the above functions, the server-aided computation can be accomplished.

Like the above example, the IC card 9 sends d' which has been obtained in the equation (58) to the terminal unit 2. The terminal unit 2 obtains M' from the following equation.

$$M'=C^{d'} \bmod n \quad (62)$$

The terminal unit 2 sends M' back to the IC card 9. The IC card 9 obtains M from M' in accordance with the M obtainment process determined by the conversion process of the equation (58). Like the above example, for the conversions according to the equations (60) and (61), along with the computation by the terminal unit 2, the following values necessary for the conversion for obtaining M from M' can be computed.

$$X_{ip}=(C \bmod p)^{r_{ip}} \bmod p \quad (63)$$

$$X_{iq}=(C \bmod q)^{r_{iq}} \bmod q \quad (64)$$

In this embodiment, the method for obtaining M is omitted because it can be easily accomplished by applying the prior art and the embodiments of the first and the second inventions.

Therefor, according to the first, the second, and the third inventions, the method for accomplishing most of the processes of the terminal unit 2 and the IC card 9 at the same time is provided and thereby the process time necessary for the server-aided computation can be remarkably reduced.

In addition, according to the first invention to the third inventions, it is not necessary to excessively increase the process speed of the IC card, thereby reducing the costs of the terminal unit 2 and the IC card 9.

In the following, an embodiment of the fourth invention is described.

This embodiment is described assuming that the IC card 9 and one POS unit are used as the main unit and the auxiliary unit, respectively.

The system structure is the same as those shown in FIGS. 2 to 4. The computation section is shown in FIG. 7.

The invention can be expressed in a general form where k D_1 and k D_2 are provided so that a modulus n can be separated into k factors each of which is a prime number. Namely, D_{j1}, D_{j2} (j = 1, 2, ... k) FIG. 7 is shown in such manner. In this embodiment, firstly, the case of k = 1 is described.

As shown in FIG. 7, the IC card 9 stores a plurality of positive integers $D_1 = [d_0, f_1, f_2, \ldots, f_m]$ and the open information storage section 1*b* stores $D_2 = [d_1, d_2, \ldots, d_m]$, each of which satisfies the following equation.

$$d = d_0 - f_1 d_1 - f_2 d_2 - \ldots - f_m d_m \pmod{\lambda(n)} \quad (65)$$

where D1 is secret information of the IC card 9 and is structured so that it cannot be normally read from the outside.

The user of the IC card 9 generates a modulo-exponentiation value expressed by the following equation from the digital information M using the IC card 9 and the terminal unit 2.

$$S = M^d \bmod n \quad (66)$$

It is a digital signature of the RSA cryptosystem.

In this embodiment, a generation of the digital signature is exemplified. The present invention is applicable also to the encipherment of the RSA cryptosystem.

The computation process is described in the following by referring to FIG. 8.

The user inserts the IC card 9 into the reader writer 7 of the terminal unit 2, commands the start of the terminal unit 2 in accordance with a predetermined sequence, and enter the message M (in steps 801 to 804).

The message is detail of shopping, for example.

The IC card 9 transfers $D_2 = [d_1, d_2, \ldots, d_m]$ and the value n of the modulus to the terminal unit 2 (in steps 805 and 824).

The terminal unit 2 computes m y_i's using D_2 and n received by computation sections $2a_1$ to $2a_m$ as expressed in the following equation.

$$y_i = M^{d_i} \bmod n \ (i = 1, \ldots, m) \quad (67)$$

After that, the terminal unit 2 transfers y_i to the IC card 9 (in steps 825 and 807).

On the other hand, the IC card 9 obtains y_0 from the following equation using the secret information d_0 in a computation section 9*c*.

$$y_0 = M^{d_0} \bmod n \quad (68)$$

In addition, the IC card 9 obtains the signature S from the following equation using $y_1, \ldots, y_m$ received from the terminal 2 (in step 808).

$$S = y_0 \cdot y_1^{f_1} \cdot y_2^{f_2} \cdot \ldots \cdot y_m^{f_m} \bmod n \quad (69)$$

The IC card 9 transfers the signature S to the terminal unit 2 (in step 809). The terminal unit 2 records the signature S and completes the signature process.

In this embodiment, when $f_1, \ldots, f_m$ are represented in binary notation (0 or 1), the computation of the power which is apparently present in the equation (69) can be omitted, thereby decreasing the computation load of the IC card 9.

There are following three major effects in this embodiment.

(1) Since the secret information of the IC card 9 is not directly transferred to the terminal unit 2, the terminal unit 2 cannot know the secret information d of the IC card 9. In addition, the secret information d can be kept secret against a third party who tries to wiretap the communication between the IC card 9 and the terminal unit 2.

(2) Using a sufficiently high speed terminal unit 2, the process time can becomes shorter than that of the IC card 9 which executes the modulo-exponentiation computation.

(3) By properly selecting the secret information of the IC card 9, the process time can be reduced. This effect becomes remarkable when $d_0 \cong 2$ and $f_1, \ldots, f_m$ are represented in binary notation. This embodiment can be also applied to the key-in-common system proposed by Diffie - Hellman. In this case, the differences are that n becomes prime number p and

$$\lambda(n) = p - 1$$

Another embodiment according to the fourth invention is described in the following.

In this embodiment, like the above embodiment, generations of the RSA cryptosystem and a digital signature are exemplified using the IC card system. It is necessary to note that in the RSA cryptosystem n is the product of two large prime numbers p and q and $n = p \cdot q$ can be satisfied.

In this embodiment, in the IC card 9, a plurality of positive integer sets $D_{11} = [d_{10}, f_{11}, \ldots, f_{1m}]$, $D_{12} = [d_{20}, f_{21}, \ldots, f_{2m}]$, and $D_2 = [d_1, \ldots, d_m]$ have been stored, each of which satisfies the following equations.

$$d = d_{10} - f_{11} d_1 - f_{12} d_2 - \quad - f_{1m} d_m \pmod{p-1} \quad (70)$$

$$d = d_{20} - f_{21} d_1 - f_{22} d_2 - \quad - f_{2m} d_m \pmod{p-1} \quad (71)$$

where D_{11} and D_{12} are secret information of the IC card 9 and they are structured so that they cannot be normally read from the outside.

The user of the IC card 9 generates a modulo-exponentiation value expressed by the following equation from the digital information M using the IC card 9 and the terminal unit 2.

$$S = M^d \bmod n \quad (72)$$

FIG. 9 shows process steps for computing the signature S in the embodiment.

Since the process steps of FIG. 9 are the same as those of FIG. 8 except for the power remainder computation section in step 940, only the step 940 is described in the following.

Firstly, the IC card 9 transfers the open information D_2 to the terminal unit 2 (in steps 905 and 934).

The terminal unit 2 computes m y_i's using D_2 and n, which have been received, from the following equation.

$$y_i = M^{d_i} \bmod n \quad (i = 1, \ldots, m) \quad (73)$$

The terminal unit 2 transfers y_i to the IC card 9 (in steps 935 and 907).

On the other hand, the IC card 9 obtains y_{10} and y_{20} from the following equations using the secret information d_{10} and d_{20} stored in the IC card 9 (in step 908).

$$y_{10} = M^{d_{10}} \bmod p \quad (74)$$

$$y_{20} = M^{d_{20}} \bmod q \quad (75)$$

The IC card 9 computes S1 and S2 using $y_1, \ldots, y_m$, which have received from the terminal unit 2, and y_{10} and y_{20} from the equations (74) and (75) (in step 908).

$$S_1 = y_{10} \cdot y_1^{f_{11}} \cdot y_2^{f_{12}} \cdots y_m^{f_{1m}} \bmod p \quad (76)$$

$$S_2 = y_{20} \cdot y_1^{f_{21}} \cdot y_2^{f_{22}} \cdots y_m^{f_{2m}} \bmod p \quad (77)$$

Since p and q are relatively prime, from the Chinese remainder theorem, the number S which satisfies the following equations and which is less than n is uniquely determined. The number S is the desired digital signature S.

$$S_1 = S \bmod p \quad (78)$$

$$S_2 = S \bmod q \quad (79)$$

Although these equations can be solved in various methods, by computing Wp and Wq which satisfy the following equations

$$W_p = q\,(q^{-1} \bmod q) \quad (80)$$

$$W_q = p\,(p^{-1} \bmod q) \quad (81)$$

and by storing them in the IC card 9 in advance, S can be computed from the following equation.

$$S = (S_1 \cdot W_p - S_2 \cdot W_q) \bmod n \quad (82)$$

The IC card 9 transfers S to the terminal unit 2 (in step 909). The terminal unit 2 records it and completes the signature generation process (in steps 936 to 938).

In this embodiment, the same effects as (1) to (3) described in the above embodiment are accomplished. Particularly, the effect (3) is remarkably accomplished when $d_{10} \cong 2$ and $d_{20} \cong 2$ and $f_{11}, \ldots, f_{1m}$ and $f_{21}, \ldots, f_{2m}$ are represented in binary notation.

Lastly, the effect of shortening the process time in another embodiment of the fourth invention is shown in FIG. 10.

The vertical axis of the chart represents a relative value of the process time (assuming that the process time on which the IC card 9 generates a signature by itself is 1). The horizontal axis of the chart represents a relative value v of the process speed of the terminal unit 2 (assuming that the computation speed of the IC card 9 is 1).

Approximately, in the range of $20 \leqq v \leqq 1000$, it is obvious that the process time of the server- aided computation method of the fourth invention becomes short.

Consequently, according to the fourth invention, the secret information of the main unit such as the IC card 9 which operates as the main computation unit can be processed in shorter time than that executed by the main unit alone using the computation capacity of the auxiliary unit without divulging the secret information to the auxiliary unit.

A method for determining whether the result obtained by the server-aided computation method described above is valid or invalid and the related units are described in the following.

FIG. 11 is a block diagram showing a system composed of the IC card 9 and the auxiliary unit. FIGS. 12 and 13 are a process flow chart embodying the fifth invention and an outline diagram showing the general structure, respectively.

In the IC card 9, decryption keys of the RSA cryptosystem have been stored. Now, assume that the keys are d, n, p, and q where p and q are large prime numbers which are kept secret to the outside except for the IC card 9. The open modulus n is the product of p and q. d is a secret exponent. The exponent e and the modulus n which structure the open keys may be open to the requested side of the computation.

The IC card 9 is provided with verification means IB

The IC card 9 requests the terminal unit 2 to execute the decryption conversion or generate a signature without divulging the secret key d. By a proper server-aided computation method, the requested side of the computation obtains the message M which has been converted and the message S of the computation result. When the conversion has been validly executed on the requested side of the computation, it is necessary to satisfy the following equation.

$$M = S^e \bmod n \quad (83)$$

To check the equation, when generating the keys, it is necessary to consider the creation method of the open exponent e.

As well known, when $L = LCM\,(p-1, q-1)$ is defined, the value of e can be freely decreased without degrading the safety, if e and L are relatively prime

The IC card 9 computes M, which satisfies the following equation using S as the result of the server-aided computation in step 1206. This computation can be executed by the IC card 9 at a satisfactorily high speed because the value of e is small.

$$M' = S^e \bmod n \quad (84)$$

Then, M' obtained in step 1207 is compared with the former message. When both of them are matched, it can be determined that the computation by the terminal unit 2 has been validly executed.

In the server-aided computation method relating to the conversion f_k according to the secret information k, when the requesting side of the computation can easily execute the reverse conversion f_k^{-1}, this method can be generally applied to any other server-aided computations as well as the RSA cryptosystem. FIG. 13 shows the system structure where the server-aided computation method is generally extended.

In FIG. 13, the requesting side of the computation adds the secret information k to the input information x in a first process section 51, computes $y = f_k(x)$ with an assistance of a process section 52 on the requested side, solves $x' = f_k^{-1}(y)$ in a second process section 53, and compares x' with the input information x in a comparison section 54 so as to verify the process result.

Therefore, in this embodiment, since the input information x and the information x' obtained by the reverse

conversion are compared in the comparison section 54, the server-aided computation can be really validated.

By referring to FIGS. 14 and 15, another embodiment of the fifth invention is described in the following. In this embodiment, a server-aided computation of a modulo-exponentiation in the Diffie - Hellman type key-in-common protocol is described as an example.

Firstly, the Diffie - Hellman type key-in-common protocol is described. When the key data is shared between the user A and the user B, the following process is executed.

As the preparation, it is necessary to generate for a user i a secret key x_i, compute the open key $p_i = g^{x_i} \bmod p$, and open p_i to the public where p is a prime number and g is a primitive root of the Galois field GF (p) which are common in all users. The user A obtains a common key K_{AB} by the computation of the equation (85) using the open key p_B of the user B and the own secret key x_A.

$$K_{AB} = p_B^{x_A} = g^{x_B x_A} \bmod p \tag{84}$$

The user B obtains the common key K_{BA} by the computation of the equation (86) using the open key p_A of the user A and the own secret key x_B.

$$K_{BA} = p_A^{x_B} = g^{x_A x_B} \bmod p \tag{86}$$

K_{BA} accords with K_{AB}. In addition, it is difficult to obtain the secret key using the open key because it requires computing a discrete logarithm.

The above key-in-common protocol can be accomplished by executing the modulo-exponentiation $p_B^{x_A} \bmod p$ where the prime number p is the modulus.

Next, the server-aided computation protocol of a modulo-exponentiation where the prime number p is the modulus is described in the following. Assume that the requesting side of the computation is the IC card 9 and the requested side is the terminal unit 2 which has higher computation capacity than the IC card 9. The IC card 9 has stored a fixed secret key x. The base g which is the open key and the modulus p which is a prime number are known by both the requesting side and the requested side of the computation.

The IC card separates the secret key x as expressed in the equation (87). This process can be executed by the IC card itself or the center as a key issuance process. In addition, it is also possible to store this process in the IC card as secret information.

$$x = x_0 + f_1 x_1 + f_2 x_2 + \ldots + f_m x_m \bmod p - 1 \tag{87}$$

where fi is 0 or 1 and x_0 is a small value.

$F = [f_1, f_2, \ldots, f_m] D = [x_1, x_2, \ldots, x_m]$, and x_0 are named separated members of the key x.

These x, F, D, and x_0 are secret information of the IC card 9. The separation method of the key x is a modification of the method of the RSA-S1 protocol (proposed in the thesis of Matsumoto and Imai, "How to ask services without violating privacy", 1989 Encipherment and Information Security Symposium Text, February 1989).

The server-aided computation protocol using the key separation is shown in FIG. 14.

(1) The IC card 9 sends D to the terminal unit 2 (in step 1401).

(2) The terminal unit 2 receives the separated member D_1 in step 1410, computes z_i ($1 \leqq i \leqq m$) of the following equation (88) in step 1411, and sends the resultant data $Z = [z_1, z_2, \ldots, z_m]$ to the IC card 9 in step 1412.

$$z_i = g^{x_i} \bmod p \tag{88}$$

(3) The IC card 9 receives Z1 in step 1402 and computes K (K_1) of the equation (89) in step 1403.

$$K = \left(\prod_{f_i = 1} z_i \bmod p \right) g^{x_0} \bmod p = g^x \bmod p \tag{89}$$

f_i is not limited to 0 and 1, but extensible to general positive integers.

In this embodiment, a method for determining whether the process of the terminal unit 2 has validly executed the server-aided computation of the modulo-exponentiation and for detecting the result if it is invalid is provided. As one example, a method for checking that signatures obtained by different separated members of keys used for the server-aided computation protocol are matched is described in the following.

(1) The IC card 9 creates two separated members which satisfy the equation (21) and names them D_1 and D_2.

(2) The IC card 9 executes the process of the server-aided computation using D and obtains the result K_1

(3) The IC card 9 executes the process of the server-aided computation using D_2 in step 1404.

(4) The terminal unit 2 receives D_2 in step 1413, computes the following equation in step 1414.

$W_i = gY_i \bmod p$ ($1 \leqq i \leqq n$), obtains the following in step 1415.

$W_2 = [W_1, W_2, \ldots W_n]$ and sends the result to the IC card 9.

(5) The IC card 9 receives W_2 from the terminal unit 2 in step 1405 and obtains the result K_2 in step 1406

(6) The IC card 9 compares K_1 with K_2 in step 1407 When they are matched, the IC card 9 determines that the computation result is valid in step 1408. When they are not matched, the IC card 9 determines that the computation result is invalid in step 1409.

In the above process, it is possible to verify whether the terminal unit 2 has validly executed the process

However, when two keys are separated, x_0 should be different between them. If they are not different, when the terminal unit 2 computes the equation (88) using g' as the base rather than g, the results K_1 and K_2 are matched.

It is obvious that the result of the computation of the equation (88) using g as the base differs from that using g' as the base. The RSA-S1 protocol, which is the original form of the above protocol, is a method equivalent to $X_0 = 0$. In the method for verifying the computation result by executing the above method twice, an attack method using g' instead of g is present. Thus, x_0 is added as a separated member of a key.

The same result can be obtained by other methods as well as the method described in this embodiment. For example, the practical protocol can be generalized by increasing the number of the separated members to 3 or more. In addition, when computing K_1 and K_2, it is also possible to use a different server-aided computation method.

FIG. 15 shows a general structure of this embodiment.

(1) Using first process sections 55 and 56 on the requesting side and the requested side, respectively, the result y_1 is obtained.

(2) Using second process sections 57 and 58 on the requesting side and the requested side, respectively, the result y_2 is obtained.

(3) By comparing y_1 with y_2 in a comparison section 59 on the requesting side, when they are matched, it is determined that the result is valid. Otherwise, it is determined that the result is invalid.

An independent server-aided computation method which can be used for such compound type protocol can be selected from those which have been proposed.

Only the server-aided computation method for computing modulo-exponentiation necessary for the Diffie-Hellman type key-in-common protocol has been described. In a general server-aided computation, the validity of the computation result can be verified by the method described above. For example, for verifying the computation result of the RSA cryptosystem, it is possible to use the same method.

By referring to FIGS. 16 and 17, another embodiment is described in the following. The concept of the method described in the following is similar to that of the above embodiment. As outlined in FIG. 16, generally, when information which has not been converted and that which has been converted are x and y, respectively, in the server-aided computation method for the conversion $y = f_K(x)$ according to secret information k, the conversion process is executed in first process sections 60 and 61. When reverse conversion f_K^{-1} is present, it is obtained by second process sections 62 and 63. A comparison section 64 verifies the fidelity of the requested side of the computation using the reverse conversion by checking that $x' = f_K^{-1}(y)$ accords with x.

The outline of the protocol is as follows.

(1) x is converted using the server-aided computation of the forward conversion and the requesting side of the computation obtains y.

(2) y is converted using the server-aided computation of the reverse conversion and the requesting side of the computation obtains x'.

(3) The requesting side of the computation compares x with x' and when they are matched, it determines that the result y is valid.

However, in the first embodiment, the reverse conversion was applicable only when it could be easily executed on the requesting side. However, in this embodiment, the reverse conversion is applicable, even if the reverse conversion cannot be executed only by the requesting side. To accomplish that, in this embodiment, the server-aided computation is also applied to the reverse conversion.

The structures of practical forward and reverse server-aided computations should be considered depending on individual applications.

It is necessary to note that the information necessary for the reverse server-aided computation is transferred to the outside of the unit on the requesting side of the computation as well as that necessary for the forward server-aided computation. Thus, the protocol should be structured in the manner that the secret information k is not divulged to the outside except for the requesting side even if the two types of information are combined. In addition, generally, the requested side sends information back to the requesting side one time each for the forward server-aided computation and the reverse server-aided computation. In total, the information is sent two times. When the information which is sent two times is not that which is not obtained in the valid process, the protocol should be structured so that the protocol does not allow the information to be passed to the last verification.

Using an example of the server-aided computation of the RSA cryptosystem, it is possible to represent that the practical protocol can be structured. The protocol that follows is available when the computation load of the reverse conversion is large, namely, the open exponent e is large.

Like the embodiments described above, assuming that the requesting side of the computation is the IC card 9 and the requested side of the computation is the terminal unit 2 which has a higher computation capacity that the IC card 9, a practical example of the process is shown in FIG. 17.

The IC card 9 has stored own fixed decipher keys d, n, p, and q of the RSA cryptosystem where p and q are kept secret to the outside except for the IC card 9, d is a secret exponent, and the exponent e and the modulus n which structure the open keys may be open to the requested side of the computation. The server-aided computation method described in the above embodiment is practically exemplified in the following.

The IC card 9 knows a random number R_0 which satisfies the following conditions and which is kept secret to the terminal unit 2.

(1) $r_p = R_0 \bmod (p-1)$,

(2) $r_q = R_0 \bmod (q-1)$, and

(3) The value of $\chi(r_p) + \chi(r_q)$ is relatively small

The IC card 9 sends d' and n which have been computed from the equation $d' = (d - R_0) \bmod L$ to the terminal unit 2 (in step 1701). The terminal unit 2 receives them in step 1713, computes the equation $S' = M^{d'} \bmod n$ in step 1715, and sends the result to the IC card 9 in step 1716. The IC card 9 receives S' in step 1704 and obtain S from $S = M^R \cdot S' \bmod n$ in step 1705 The above process is the forward server-aided computations.

On the other hand, the IC card 9 has computed $Q = R_1^d \bmod n$ using another random number R_1 and the modulus n. The computation load for computing Q_1 from R_1 is large. However, to reduce the computation load, it is possible to compute Q_1 in advance using a non-busy time of the CPU 15 of the IC card 9. When a plurality of random numbers and sets of their powers are generated in the non-busy time, signatures can be successively generated (or cipher texts can be successively deciphered). The set of R_1 and Q_1 can be also generated in the manner that firstly $R_1 = Q_1^e \bmod n$ and Q_1 have been generated and then R_1 is assigned.

A procedure for the reverse server-aided computation for determining the validity of S obtained as the result of the forward server-aided computation is described in the following.

(1) The IC card 9 computes the product of S and Q1, which is $Z = (S \cdot S1) \bmod n$ in step 1706 and sends Z to the terminal unit 2 in step 1707.

(2) The terminal unit 2 receives Z in step 1717, computes $W = Z^e \bmod n$ using the open exponent e in step 1718, and sends the result to the IC card 9 in step 1719

(3) The IC card 9 receives the result in step 1708 and computes $V = (W / R1) \bmod n$ in step 1709.

(4) The IC card 9 compares V with M in step 1710 and when they are matched, it determines that the result S of the forward server-aided computation is valid.

In the following, the reason why the steps (1) to (4) above allows the validity of the forward server-aided computation to be determined is described.

In the step (4) of the reverse server-aided computation, to allow V to be matched with M, $V=(W / R1) \bmod n$ should be satisfied, thereby $W=M \cdot R1 \bmod n$. When $W=M \cdot R1 \bmod n$, it is determined that all the server-aided computation failed. In this protocol, the terminal unit 2 knows M. Thus, even if the forward server-aided computation failed, when the terminal unit knows R1 in the step (2), it may compute $W=M \cdot R1 \bmod n$ and cause the IC card 9 to generate an invalid signature. However, since the terminal unit 2 knows R1 only when it correctly executes the forward server-aided computation, the validity of the forward server-aided computation can be determined in the above process.

By referring to FIG. 18, a third embodiment is described in the following. This embodiment can be applied to verify a signature of the RSA cryptosystem being generated. The concept is that the verification of the forward server-aided computation is performed using the server-aided computation of the reverse conversion like the embodiment shown in FIG. 12.

The IC card 9 has obtained u, v, and w which satisfy the equation (90) using the open exponent e in advance.

$$e=u \cdot v-w \tag{90}$$

(1) The IC card 9 requests the terminal unit 2 to generate a signature without divulging the secret key d to the terminal unit d. At the time, the parameter w of the equation (90) should satisfy the following two conditions. (Condition 1) $W \neq 0$ (Condition 2) e·d'0 are not divided by w.

(2) The IC card 9 sends S which has been obtained as the result of the server-aided computation for generating the signature in (1) to the terminal unit 2 in step 1806.

(3) The terminal unit 2 receives S in step 1816, obtains U of the equation (91) where S is raised to the u-th power in step 1817, and sends U to the IC card 9 in step 1818.

$$U=S^{u} \bmod n \tag{91}$$

(4) IC card 9 receives U in step 1807 and computes V of the following equation (92) in step 1808.

$$V=U^{v} \cdot S^{w} \bmod n \tag{92}$$

(5) The IC card determines whether V and the plain text M are matched in step 1809. When they are matched, the IC card 9 determines that S is the valid signature. Otherwise, the IC card 9 determines that invalid computations have been executed.

Consequently, since $V=S^{uv-w}=S^{e} \bmod n$ is satisfied, when the terminal unit 2 has correctly executed the computation, $S=M^{d} \bmod n$ is obtained, thereby $V=S^{e}=M^{de}=M$.

If the terminal unit 2 has not validly executed the computation in the step (1) above and it has obtained S is not valid S, it is necessary to consider whether the terminal unit 2 can obtain U which is passed only to the last verification.

$M=U^{v} \cdot S^{w} \bmod n$ is the last verification equation. Although S^{w} can be obtained by the terminal unit 2, it is necessary to lastly obtain U which satisfies $U^{v}=M / S^{w} \bmod n$. Namely, v-th root in the modulus n should be obtained. Consequently, it is difficult to obtain U which can pass only the last verification equation.

When the two conditions for W have not been satisfied, it is possible to change the structure in the manner that only the last verification equation is passed without obtaining the v-th root in the modulus n.

In addition, in the steps (3) to (5), the secret information of the IC card 9 has not been used. Thus, unless the secret information of the IC card 9 is divulged by the server-aided computation protocol for generating the signature used in the step (2), the secret information is never divulged. Since it is possible to consider that only the step (2) prevents the secret information from divulging, the secret information is never divulged through these steps.

In these steps, $V=3$ and $W=2$ can be set depending on the value of the open key e. In this case, the computation amount that the IC card 9 executes becomes minimum and the modulo-multiplication is executed four times. Thus, when the value of the open key e is large, the process time can be beneficially reduced.

By referring to FIG. 19, another embodiment is described in the following.

The IC card 9 has created secret information t which is used in the server-aided computation for the reverse conversion in advance. Although t is a random number, it is restricted to the condition where the value of $\chi(t_p)-\chi(t_q)$ is small as to effectively execute the verification where t_p and t_q are values defined in the following equations (93) and (95). These conditions are the same as those used in the server-aided computation of the above embodiment.

$$t_p=t \bmod (p-1) \tag{93}$$

$$t_q=t \bmod (q-1) \tag{95}$$

As shown in FIG. 19, assume that the IC card 9 and the terminal unit 2 have obtained the signature text S in accordance with steps 1901 to 1905 and steps 1912 to 1915, respectively. To verify the validity of the signature text S, the IC card 9 and the terminal unit 2 execute steps 1906 to 1909 and steps 1916 to 1918, respectively.

(1) The IC card 9 computes $Y=S^{t} \bmod n$ in step 1906 and transfers Y being computed to the terminal unit 2.

(2) The terminal unit 2 receives Y in step 1916, computes $Z=Y^{e} \bmod n$ using the open keys e and n in step 1917, and transfers Z being computed to the IC card 9 in step 1918.

(3) The IC card 9 computes $W=M^{t} \bmod n$ in step 1907 and receives Z being computed in step 1918 from the terminal unit 2 in step 1908.

(4) The terminal unit 2 determines that $Z=W$ in step 1909 and advances to step 1901. When determined that $Z \neq W$ in step 1909, the terminal unit 2 determines that the steps being executed are invalid, advances to step 1910, and informs the user of the invalidity.

The modulo-multiplication method of the steps (1) and (3) is supplemented in the following. Although $Y=S^{t} \bmod n$ is computed in the modulus n, the multiplication should be executed approximately $\log_2 t$ times. When the equation is computed by dividing it into two modulo-exponentiations relating to two prime numbers p and q structuring the modulus n according to the

Chinese remainder theorem, the computation time can be reduced.

In addition, the number of times of computing the multiplication in the modulus p is $\chi(t_p)$ and that in the modulus q is $\chi(t_q)$. When t is selected, if the condition where the value of $\chi(t_p)+\chi(t_q)$ is small has been imposed, the computations of the modulo-exponentiations of the steps (1) and (3) can be effectively executed by a unit with small computation capacity such as the IC card 9.

When t is selected to a small value, although the computation load of the IC card 9 is reduced, the IC card 9 becomes weak against attacking the estimation of t in the round robin method. Thus, it is necessary to increase the value of t to some extent.

In the embodiment in FIG. 19 shows a case where this verification method is associated with a special server-aided computation. However, this verification method is not limited to the special server-aided computation method.

In this verification method, when the open exponent e is a composite number, the reliability of the verification may degrade.

For example, assume that e is the product of two integers a and b, namely, $e=a\cdot b$.

If the terminal unit 2 invalidly changes the server-aided computation result S to $S'=S^a \bmod n$, the terminal unit 2 correspondingly computes $Z'=Y^b \bmod n$ in the step (2) of the above verification rather than computing $Z'=Y^b \bmod n$. In this case, since Z' that the IC card 9 has obtained becomes M' from the deformation of the following equation, the terminal unit 2 succeeds in passing the verification.

$$\begin{aligned} Z' &= Y^b \bmod n \\ &= S'^b \bmod n \\ &= S^{ab} \bmod n \\ &= (S^{ab})^t \bmod n \\ &= S^{et} \bmod n \\ &= M^t \bmod n \end{aligned}$$

This attack method succeeds only when e is a composite number, the factorization in prime factors is known, and the result of the server-aided computation can be changed so that $S'=S^a \bmod n$ is satisfied. Thus, when a prime number is selected for e, this attack method fails and the verification becomes effective. Even if e is a prime number, the degree of safety of the RSA cryptosystem does not degrade.

Next, another embodiment is described in the following.

The protocol that follows is valid only when the computation load in the reverse conversion is large to some extent, namely, the value of the open exponent e is large. This method is used to prevent the Hastad's attack in simultaneous transmission (J. Hasted, "On using RSA with low exponent in a public key network", Crypto 85, pages 403–408, 1985), to increase the value of e more than that of logn as to allow any plain text to be folded in the modulus n more than one time, and to become the open exponent e common in all the users by defining the quartic Fermat's number ($=2^{16}+1$) to e.

By referring to FIG. 20, a protocol which uses a server-aided computation method which is a modification of the RSA-S2 protocol (proposed in the thesis of Matsumoto and Imai, "How to ask services without violating privacy", 1988 Encipherment and Information Security Symposium Text, February 1988) is described in the following.

[1]The IC card 9 obtains the converted result S of the plain text M in accordance with the following protocol without transferring the secret key d to the terminal unit 2.

(1) The IC card 9 separates the secret key d as expressed in the following equations.

$$d=d_{op}-f_1d_1-f_2\cdots-f_md_m \bmod p-1$$

$$d=d_{oq}-g_1d_1-g_2\cdots-g_md_m \bmod q-1$$

$$D=[f_1f_2\cdots d_m].$$

$$F=[f_1f_2\cdots f_m].$$

$$G=[g_1g_2\cdots g_m].$$

where

F and G are binary values. (Generally, f_i and g_i can be positive integers) However, the expression Weight (F) $-$ Weight (G) $-\chi(d_{op})-\chi(d_{oq})\leqq L$ should be satisfied (where L is a parameter which is determined by the degree of safety).

d, d_{op}, F, and G are secret information of the IC card 9. As described below, since the terminal unit 2 cannot know d_{op}, d_{oq}, F, and G via the protocol, it cannot obtain the secret information d.

(2) The IC card 9 sends the modulus n and D to the terminal unit 2 (in step 2001).

(3) The terminal unit 2 computes the following equation and sends the plain text M and Z to the IC card 9 in step 2014.

$$Z_i=M^{d_i} \bmod n \ (1\leqq i\leqq m)$$

$$Z=[z_1z_2\cdots z_m]$$

(4) The IC card 9 computes S_p and S_q of the following equations in step 2004.

$$S_p=\left(\prod_{f_i=1} z_i \bmod q\right)\cdot M_{op}^{d} \bmod p$$

$$S_q=\left(\prod_{g_i=1} z_i \bmod q\right)\cdot M_{oq}^{d} \bmod q$$

By combining S_p and S_q using the Chinese remainder theorem (CRT), the result S is obtained.

[2] The IC card 9 separates the open key e as expressed by the following equation.

$$e=2\cdot e'+1$$

[3] The IC card 9 computes U of the following equation and sends U and e' to the terminal unit 2 in step 2005.

$$U=S^2 \bmod n$$

[4] The terminal unit 2 computes V of the following equation in step 2016 and sends V to the IC card 9 in step 2017.

$V = U^{e'} \bmod n$

[5]The IC card 9 computes W of the following equation in step 2008.

$W = S \cdot V \bmod n$

[6]When W and M are matched in step 2009, the IC card 9 determines that S is a valid signature. When they are not matched, the IC card 9 detects in what part of the protocol an invalid process has been executed.

In evaluating the safety of the above method, when the terminal unit 2 has validly executed the computation, it is obvious that the IC card 9 determines that "the terminal unit 2 has validly executed the computation."

Then, it is necessary to consider whether the terminal unit 2 can pass the last verification and change S to an invalid signature or not. The verification equation of this embodiment is expressed as follows.

$M = S \cdot V \bmod n$

Generally, to obtain V which satisfies the above equation, it is necessary to obtain the result S of the server-aided computation [1]. However, since the result S which has been raised to the second power is sent back, it is difficult to obtain S from the value being received. Although it may be possible to obtain the result S along with S^2 which is sent back [3]by properly selecting Z which is sent to the client in [1]- (3), it has not been known, thus far. The secret information which is newly added to the server- aided computation protocol for the verification is only the result S of the server-aided computation method [1]. All other information can be obtained by the terminal unit 2 alone. In the server-aided computation method [1], even if the result S is open to the public, it seems that the secret keys of the clients may be not divulged. Thus, it is supposed that the secret keys will not be divulged via [1]to [6].

On the other hand, in this embodiment, the computation load can be generalized by separating the open key e into the form of $u \cdot e' + v$. However, as described above, since e is an odd number, it is possible to set $u = 2$ and $v = 1$. In this case, the computation load of the IC card 9 necessary for the verification becomes minimum, namely, only twice multiplication in the modulus n. The communication data amount is around 1024 bits. Thus, for example, when e is a quartic Fermat's number, if the communication time and the server's process time are ignored, a high speed verification which is approximately 8 times that in the direct method can be accomplished.

In this embodiment, the feature is that the result S of the server-aided computation [1]is not transferred to the terminal unit 2. However, depending on the server-aided computation protocol type used in [1], it is possible to pass the verification protocol [2]to [6]described above. In other words, when the KS method described in the thesis "Secret Conversion of RSA Cryptosystem Using Server-Aided Computation" (1989 Encipherment and Information Security Symposium Text, February 1989), the method described in the thesis "How to ask services without violating privacy", (1989 Encipherment and Information Security Symposium Text, February 1989), or the RSA-S1/S2 protocol (ditto) is used in the server-aided computation section, if the terminal unit 2 can successively generate information to be sent back to the IC card 9 in (1), it can know the result S of (1) by using U of (3). (In the general form of $e = u \cdot e' + v$, when $(e, u) = 1$, for protocols except for the KS method, the signature can be stolen.) When the modified method of the S2 protocol is used, the same type of the attack method has not been known.

The modified method of the S2 protocol described in this embodiment contains both the KS method and the S2 method as a special case. In other words, the separation method of the secret information in the modified method accords with the S2 method when $d_{op} = d_{oq} = 0$ is satisfied in the following equations.

$$d = d_{op} - f_1 d_1 - f_2 d_2 - \cdots - f_m d_m \bmod p - 1$$

$$d = d_{oq} - g_1 d_1 - g_2 d_2 - \cdots - g_m d_m \bmod p - 1$$

On the other hand, when $f_i = g_i = 1$ is satisfied and other f_i and g_i are all 0, the modified method becomes the KS method. However, as described above, in the KS method, there is an attack method where a random result of the server-aided computation is sent back and the last verification equation is passed and thereby the terminal unit 2 can steal the signature.

Against the modified method, the same type of the attack method has not been known. Thus, it can be said that the modified method of this embodiment is superior to the KS method and the S2 method when also considering the verification.

From the fact described above, the meaning of the separation method of the secret information in the modified method can be explained. The terms d_{op} and d_{oq} prevent the signature from being stolen. The terms $f_1 d_1 + f_2 d_2 + \cdots + f_m d_m$ and $g_1 d_1 + g_2 d_2 + \cdots + g_m d_m$ prevent the attack method where a random result of the server-aided computation is returned and the last verification is passed.

In addition, in the modified method, the number of variables is greater than those of the KS method and the S2 method (in the KS method, two variables d_{op} and d_{oq} are used; in the S2 method, tow vector variables F and G are used; while in the modified method, four variables d_{op}, d_{oq}, F, and G are used). Thus, the parameters can be more flexibly selected depending on the process speeds of the IC card 9 and the terminal unit 2 than those of other methods. Assuming that the communication time between the IC card 9 and the terminal unit 2 can be ignored, when the process speed of the terminal unit 2 is very fast, in the S2 method, the process time can becomes the shortest. When the process time of the terminal unit 2 is relatively slow, the process time of the KS method is the shortest in these methods. The modified method described in the embodiment is in the middle position of the above two methods.

Then, an embodiment of the seventh invention is described in the following.

As described above, when the requested side of the computation has validly executed the conversion, the following equation (95) should be satisfied for M and S.

$$M = S^e \bmod n \tag{95}$$

The IC card 9 separates the secret key d as expressed in the equations (96) and (97). In addition, the IC card 9 computes the equations $I = [i_1, i_2, \ldots, i_M]$ and $J = [j_1, j_2, \ldots, j_M]$ which satisfy the equations (98) and (99). This process can be executed by the IC card 9 or the center as the issuance process of the key or they can be also secretly stored in the IC card 9.

$$d = d_{op} - f_1 d_1 - \ldots - f_m d_m \bmod (p-1) \quad (96)$$

$$d = d_{oq} - g_1 d_1 - \ldots - g_m d_m \bmod (q-1) \quad (97)$$

$$1 = h_{op} - i_1 d_1 - \ldots - i_m d_m \bmod (p-1) \quad (98)$$

$$1 = h_{oq} - j_1 d_1 - \ldots - j_m d_m \bmod (q-1) \quad (99)$$

where d_{op}, d_{oq}, h_{op}, and h_{oq} are small values. $D = [d_1, d_2, \ldots, dm]$, $F = [f_1, f_2, \ldots, f_m]$, $G = [g_1, g_2, \ldots g_m]$, $I = [i_1, i_2, \ldots, i_m]$, $J = [j_1, j_2, \ldots, j_m]$, d_{op}, d_{oq}, h_{op}, and h_{oq} are named separated members of the key d. These d, G, F, I, J, d_{op}, d_{oq}, h_{op}, and h_{oq} become the secret information of the IC card 9.

The server-aided computation protocol including the verification function using the key separated members (signature generation using the RSA cryptosystem) is shown in FIG. 21.

(1) The IC card 9 sends D to the terminal unit 2 (in step 2101).

(2) The terminal unit 2 receives the separated members in step 2110, computes $Z_i = M^{d_i} \bmod n$ $(1 \leqq i \leqq m)$ in step 2111, and sends the result $Z = [z_1, z_2, \ldots, z_m]$ to the IC card 9 in step 2112.

(3) The IC card 9 receives Z in step 2103 and computes the following equation using the separated members F, G, d_{op}, and d_{oq} in step 2104. When the terminal unit 2 has not committed an invalidity, the result of the computation is expressed as follows.

$$S_p = \left(\prod_{i=1}^{m} z_i^{f_i} \bmod p \right) \cdot M^{d_{op}} \bmod p$$

$$= M^d \bmod p$$

$$S_q = \left(\prod_{i=1}^{m} z_i^{g_i} \bmod q \right) \cdot M^{d_{oq}} \bmod q$$

$$= M^d \bmod q$$

Then, by using the Chinese remainder theorem, with S_p and S_q, the signature is obtained.

When the values of f_i and G_i are limited to 0 and 1, the computation of

$$\prod_{i=1}^{m} z_i^{f_i} \bmod q$$

can be executed without using powers.

Then, a method for determining whether the terminal unit 2 has validly executed its process as to determine the validity of the signature S is described in the following.

(1) In step 2105, the IC card 9 computes the following equation using Z, which has been received from the terminal unit 2 in step 2103, and the separated members I and J. When the terminal unit 2 has not executed an invalid computation, the result is expressed as follows.

$$W_p = \left(\prod_{i=1}^{m} z_i^{i_i} \bmod p \right) \cdot M^{h_{op}} \bmod p$$

$$= M \bmod p$$

$$W_q = \left(\prod_{i=1}^{m} z_i^{j_i} \bmod q \right) \cdot M^{h_{oq}} \bmod q$$

$$= M \bmod q$$

Then, using the Chinese remainder theorem with W_p and W_q, W is obtained.

(4) The IC card 9 compares M with W in step 2106. When they are matched, the IC card 9 determines the validity of the signature S (in step 2107). When they are not matched, the IC card 9 determines that S is invalid (in step 2108). In the above process, the IC card 9 can determine whether the signature S has been generated by the valid process of the terminal unit 2. When the signature S has been generated in the valid process of the terminal unit 2, the IC card 9 has place trust in the validity of S.

In evaluating the safety of the above process method, when the terminal unit 2 has validly executed the computation, it is obvious that the IC card 9 determines that "the terminal unit 2 has validly executed the computation."

Then, it is necessary to consider the possibility of a case where the terminal unit 2 can pass the last verification equation and change the result S to an invalid signature. Assume that $W = M$ is the last verification equation. W is generated by using the secret information I, J, p, q, h_{op}, and h_{oq} which are known only by the IC card 9 in step 2105 in FIG. 21. The terms h_{op} and h_{oq} serve to prevent an attack method for passing the last verification equation when the terms of $i_1 d_1 - \ldots - i_m d_m$ and $j_1 d_1 + \ldots + j_m d_m$ are computed using a random server-aided computation result. Thus, it is difficult for the terminal unit 2 to obtain the separated member Z which passes the verification equation when a signature which is generated using Z which has been received by the terminal unit 2 is invalid.

In addition, the terminal unit 2 cannot know d_{op}, d_{oq}, F, and G via the protocol, it is impossible for the terminal unit 2 to know the secret information d.

The above separated members can be categorized as the following (1) to (4) depending on whether or not there are common portions between F and G and between I and J.

(1) When the term h+1 to the term k of F and G are the same as those of I and J, respectively:

$$F = (f_1, \ldots f_h, f_{h+1}, \ldots, f_k, 0, \ldots, 0)$$

$$G = (g_1, \ldots g_h, g_{h+1}, \ldots, g_k, 0, \ldots, 0)$$

$$I = (0, \ldots, f_{h+1}, \ldots, f_k, i_{k+1}, \ldots, i_m)$$

$$J = (0, \ldots, g_{h+1}, \ldots, g_k, j_{k+1}, \ldots, j_m)$$

(2) When the first k terms of F and I are the same as those of G and J, respectively and the values of the term k+1 to the term m of F and G are all 0:

$$F = (f_1, \ldots f_k, 0, \ldots, 0)$$

$$G = (g_1, \ldots g_k, 0, \ldots, 0)$$

$I=(f_1, \ldots, f_k, i_{k+1}, \ldots, i_m)$

$J=(g_1, \ldots, g_k, j_{k+1}, \ldots, j_m)$

(3) When the first k terms of F and I are same as those of G and J, respectively and the values of the term k+1 to the term m of I and J are all 0:

$F=(f_1, \ldots, f_k, f_{k+1}, \ldots, f_m)$

$G=(g_1, \ldots, g_k, g_{k+1}, \ldots, g_m)$

$I=(f_1, \ldots, f_k, 0, \ldots, 0)$

$J=(g_1, \ldots, g_k, 0, \ldots, 0)$

(4) When there are no same terms between F and G and between I and J:

$F=(f_1, \ldots, f_k, 0, \ldots, 0)$

$G=(g_1, \ldots, g_k, 0, \ldots, 0)$

$I=(0, \ldots, 0, i_{k+1}, \ldots, i_m)$

$J=(0, \ldots, 0, j_{k+1}, \ldots, j_m)$

Then, consider the method for generating the separated members of the above (1) to (4).

(1) is a general form which contains both common terms and non-common terms between F and G and between I and J. In this method, since the information necessary for the server-aided computation and the verification is transferred in one communication session, only the requesting side of the computation knows what terms are used for the verification. Although all the terms of Z necessary for generating the signature are not checked (in other words, when I and J are interpolated with Z, the product of the terms whose value is 0 in I and J and the corresponding terms in Z is 0. Thus, even if the terms in Z are invalidly changed, it cannot be determined), this method can satisfactorily prevent a sharpshooting which passes the verification equation even if zi is invalid by combining two pieces of information, for example, by changing terms which are not verified into invalid values or by separately sending and receiving the information for the server- aided computation and the information for the verification.

(2) allows all the terms of Z necessary for generating the signature to be checked, thereby preventing the sharpshooting described above in this separation method.

In (3), W which is compared with M for checking the validity is present during generating the signature. Thus, if Z is an invalid result, it is possible to check the validity of Z before obtaining the signature S. In addition, since an invalid signature is not generated, the computation load can be reduced. When the separated members are generated in such a manner, although all the terms of Z necessary for generating the signature are not checked, this method according to the present invention allows the sharpshooting described in (1) above to be satisfactorily prevented.

In (4), although all the terms are not checked, the method according to the present invention allows the sharpshooting described in (1) above to be satisfactorily prevented.

FIG. 22 shows a system structure which is generally extended.

In FIG. 22, the requesting side of the computation adds the secret information k to the input information x in a first process section 71, executes the pre-process of $y=f_k(x)$ and $x'=g_k(x)$ with an assistance of a process section 74 of the requested side, and obtains the process result in a third process section 75. After that, the requesting side executes the post-process of $x'=g_k(x)$ to obtain x in a second process section 72, compares x' with x in a comparison section 73, and then verifies the process result obtained in the third process section 75.

Thus, in the seventh invention, the comparison section 73 compares the input information x with the information x' obtained in the second process section 72 and checks the computation result.

Consequently, according to the fifth, sixth, and seventh inventions, when the conversion relating to secret information is executed separately by a plurality of units, the secret information is not divulged to other than a specific unit. In addition, the conversion is executed with an assistance of the computation capacity of the requested side. Moreover, since a disturbance of the computation committed by the requested side and/or the third party can be detected, the server-aided computation relating to the secret information can be much precisely executed.

What is claimed is:

1. A server-aided computation method for computing d-th power of integer C modulo n using a main unit for executing said computation with secret key d and at least one auxiliary unit for supporting a computation that said main unit executes, said method comprising the steps of:

generating d' from a secret key d using m random numbers R_i (where $i=1, \ldots, m$) generated by said main unit having secret keys n and d;

transferring d' and n from said main unit to said auxiliary unit;

computing the following equation from a message block C in said auxiliary unit

$$M=C^{d'} \bmod n$$

computing X using said random numbers R_i and n in said main unit while computing M' in said auxiliary unit;

transferring M' from said auxiliary unit to said main unit; and

computing a message block M using the following equation in said main unit

$$M=M' \cdot X \bmod n.$$

2. A server-aided computation method for computing d-th power of integer C modulo n using a main unit for executing said computation with secret key d and at least one auxiliary unit for supporting a computation that said main unit executes, said method comprising the steps of:

generating d' from a secret key d using m random numbers R_i (where $i=1, \ldots, m$) generated by said main unit having secret keys n and d;

transferring d' and n from said main unit to said auxiliary unit;

computing the following equation from a message block C in said auxiliary unit

$$M=C^{d'} \bmod n$$

computing X^{-1} using said random numbers R_i and n in said main unit while computing M' in said auxiliary unit;

transferring M' from said auxiliary unit to said main unit; and

computing a message block M using the following equation in said main unit

$$M = m \cdot X^{-1} \bmod n .$$

3. A server-aided computation method for computing d-th power of integer C modulo n using a main unit for executing said computation with secret key d and at least one auxiliary unit for supporting a computation that said main unit executes, said method comprising the steps of:

generating d' from a secret key d using m random numbers R_i (where $i = 1, \ldots, m$) generated by said main unit having secret keys n and d;

transferring d' and n from said main unit to said auxiliary unit;

computing the following equation from a message block C in said auxiliary unit

$$M' = C^{d'} \bmod n$$

computing X and X^{-1} using said random numbers R_i and n in said main unit while computing M' in said auxiliary unit;

transferring M' from said auxiliary unit to said main unit; and

computing a message block M using M', X, and X^{-1}.

4. A server-aided computation method for executing a computation to raise a positive integer M to the d-th power modulo n, using a main unit which has secret information d and at least one auxiliary unit for supporting said computation, said method comprising the steps of:

(a) decomposing said integer n into k ($k \geq 1$) positive factors n_j (where $j = 1, \ldots, k$), which are relatively prime to each other;

(b) decomposing said positive integer d into $(m+1)$ k secret integers $D_{1j} = [d_{j0}, f_{j1}, f_{j2}, \ldots, f_{jm}]$ (for $j = 1, \ldots, k$) stored in said main unit and $m \times k$ public integers $D_{2j} = [d_{j1}, d_{j2}, \ldots, d_{jm}]$ ($j = 1, \ldots, k$)

which satisfy the following k sets of equations

$$d = d_{j0} - f_{j1} \cdot d_{j1} - f_{j2} \cdot d_{j2} - \ldots - f_{jm} \cdot d_{jm} \pmod{\lambda(n_j)}$$

where $j = 1, \ldots, k$ and $\lambda(n_j)$ is the Carmichael function of said positive integer n_j;

(c) computing in said auxiliary unit $Y_{ji} = M^{d_{ji}} \bmod n$ where $i = 1, \ldots, m$ and $j = 1, \ldots, k$ and sending the results to said main unit; and

(d) computing in said main unit the following k values S_j using $Y_{j0} = M^{d_{j0}} \bmod n$ and Y_{ji} which have been computed by said main unit;

$$S_j = Y_{j0} \cdot Y_{j1}^{f_{j1}} \cdot Y_{j2}^{f_{j2}} \cdots Y_{jm}^{f_{jm}} \bmod n_j$$

where $j = 1, \ldots k$ and

(e) obtaining a result S which satisfies the k simultaneous equations concerning S as follows:

$$S_j = S \bmod n_j \text{ (for } j = 1, \ldots, k)$$

5. The server-aided computation method of claim 4 wherein said integer n is a prime number.

6. The server-aided computation method of claim 4 wherein said integer is a product of two prime numbers.

7. The server-aided computation method of claim 4 wherein each of $m \times k$ non-negative integers $f_{j1}, f_{j2}, \ldots f_{jm}$ is 1 or 0.

8. The sever-aided computation method of claim 4 wherein values d_{ij} are defined so that they satisfy a condition $d_{ij} = d_{uv}$ at least for one set of integer pairs (i, j) and (u, v) where i, j, u and v are arbitrary subscripts

9. A distributed information processing unit having a main unit for storing secret information and at least one auxiliary unit for supporting a transformation that said main unit executes, for executing a distributed process without disclosing said secret information necessary for said transformation to other than said main unit, said distributed information processing unit comprising

transformation means for transforming input information inverse transformation means for inversely transforming the transformation results, and

verification means for comparing the inverse transformation results of said inverse transformation means with said input information so as to verify the transformation results of said transformation means, said main unit having said verification means.

10. A distributed information processing unit having a main unit for storing secret information and at least one auxiliary unit for supporting a transformation that said main unit executes, for executing a distributed process without disclosing said secret information necessary for said transformation to other than said main unit, said distributed information processing unit comprising

a plurality of transformation means for executing said transformation of input information; and

verification means for mutually comparing the transformation results of said plurality of transformation means, said main unit having said verification means.

11. The distributed information processing unit of claim 9 wherein said secret information is not disclosed to said auxiliary unit and distributively processed in said main unit and said auxiliary unit.

12. A distributed information processing unit having a main unit for storing secret information and at least one auxiliary unit for supporting a transformation that said main unit executes, for executing a distributed process without disclosing said secret information necessary for said transformation to other than said main unit, said distributed information processing unit comprising:

first transformation means for executing said transformation of input information;

second transformation means for executing an identity transformation; and

comparison means for comparing the transformation results of said second transformation means with said input information so as to verify the transformation results of said first transformation means, said main unit having said comparison means.

* * * * *

United States Patent [19]

Miyaguchi

[11] **Patent Number: 4,514,592**

[45] **Date of Patent: Apr. 30, 1985**

[54] **CRYPTOSYSTEM**

[75] Inventor: **Shoji Miyaguchi, Yokohama, Japan**

[73] Assignee: **Nippon Telegraph & Telephone Public Corporation, Tokyo, Japan**

[21] Appl. No.: **398,016**

[22] Filed: **Jul. 14, 1982**

[30] **Foreign Application Priority Data**

Jul. 27, 1981 [JP] Japan 56-117499
Oct. 19, 1981 [JP] Japan 56-166777
Jun. 21, 1982 [JP] Japan 57-106567

[51] **Int. Cl.**[3] **H04L 9/04**
[52] **U.S. Cl.** **178/22.11; 178/22.14**
[58] **Field of Search** 178/22.11, 22.14

[56] **References Cited**

U.S. PATENT DOCUMENTS

4,200,770 4/1980 Hellman et al. 178/22.11
4,351,982 9/1982 Miller et al. 178/22.11
4,405,829 9/1983 Rivest et al. 178/22.11

Primary Examiner—Sal Cangialosi
Attorney, Agent, or Firm—Pollock, Vande Sande and Priddy

[57] **ABSTRACT**

A cryptosystem for the RSA cryptography which calculates $C \equiv M^e \bmod n$ and, for this calculation, performs an operation $C = M_1 \times M_2 \bmod n$. An operation

$$R_j = M_1 \times M'_{2j} + 2^\lambda R_{j+1} - \left[\frac{M_1 \times M'_{2j} + 2^\lambda R_{j+1}}{n} \right] \times n,$$

where

$$M'_{2j} = M_{2j} - \omega\delta_{j\lambda} \cdot 2^\lambda + \omega\delta_{(j-1)\lambda},$$

$$M_{2j} = \sum_{i=0}^{\lambda-1} \delta_{(j-1)\lambda+i} \cdot 2^i,$$

$$M_2 = \sum_{i=0}^{n\lambda-1} \delta_i \cdot 2^i, \ \delta_i = 0 \text{ or } 1,$$

$$R_{l+1} = 0, \text{ and } \omega = 0 \text{ or } 1,$$

is performed in the order $j=l, l-1, \ldots 1$ to obtain last R_1 as the result of the calculation $M_1 \times M_2 \bmod n$. The calculation

$$\left[\frac{M_1 \times M'_{2j} + 2^\lambda R_{j+1}}{n} \right] = Q_j$$

is performed in a quotient calculating unit, and the calculation $M_1 \times M_{2j} + 2^\lambda R_{j+1} - Q_j n$ is performed in a main adding unit. Where, variable R_j may be divided into two parts $R_{j,0}$ and $R_{j,1}$. In this way, the multiplication and the division are simultaneously conducted, thereby to raise the calculation speed.

22 Claims, 186 Drawing Figures

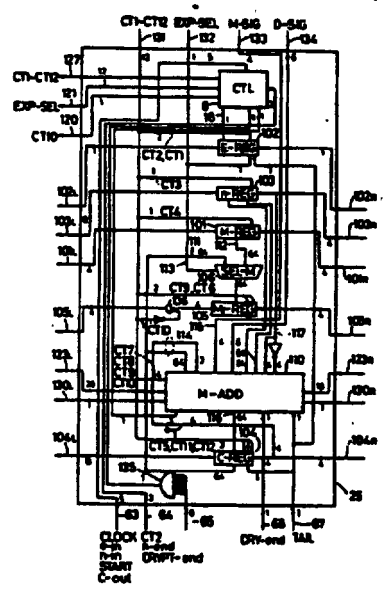


FIG. 1 PRIOR ART

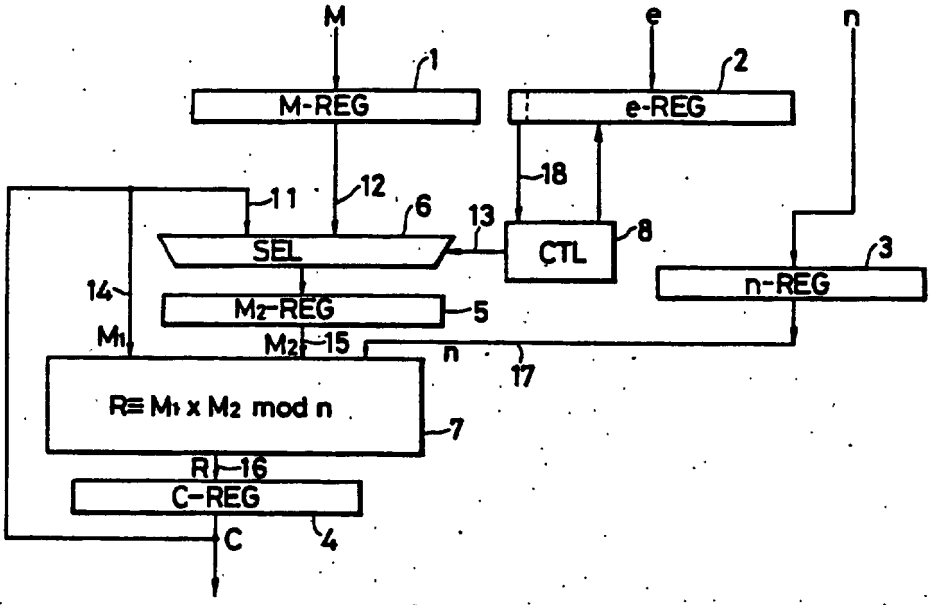


FIG. 2

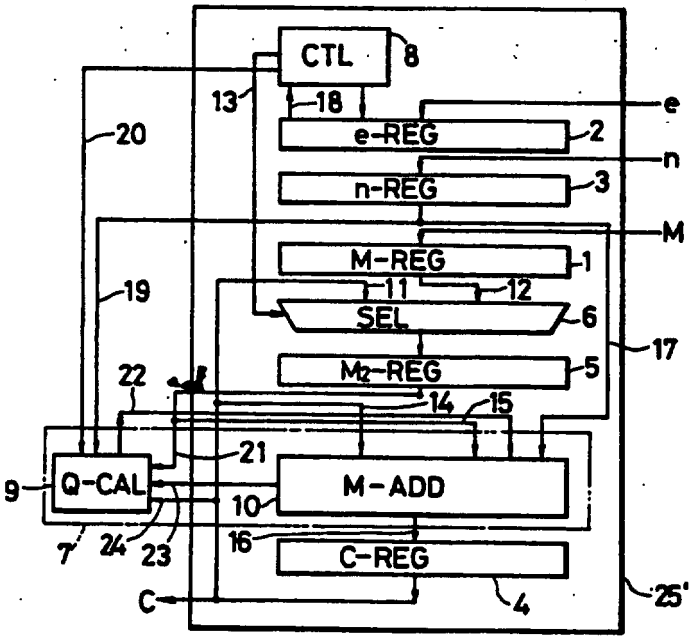


FIG. 3

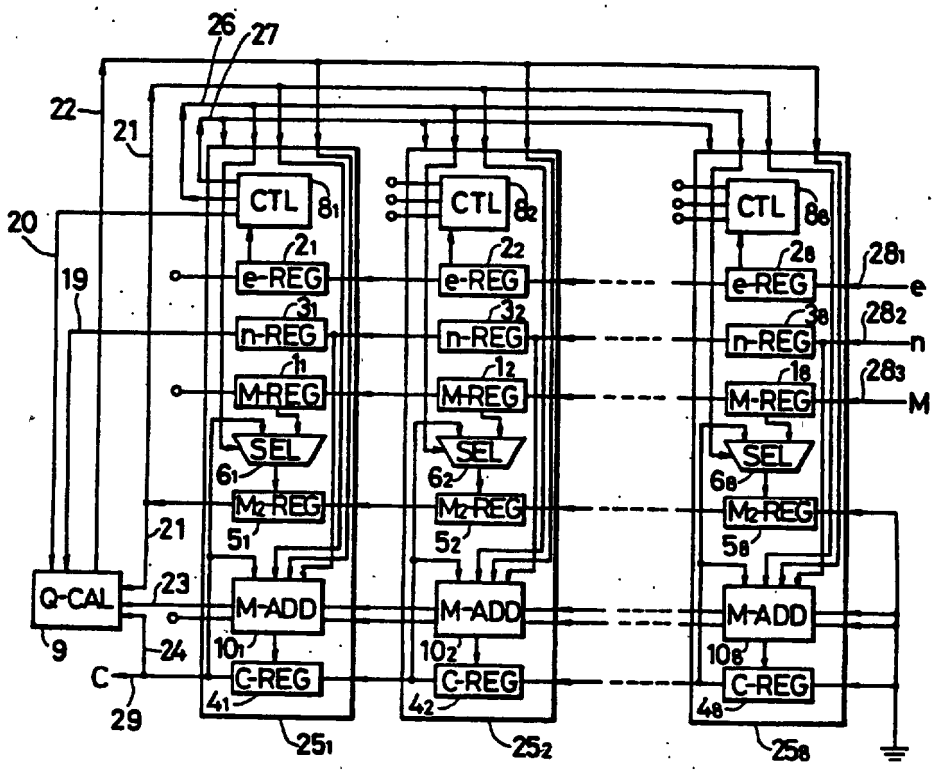


FIG. 4A

FIG. 4B

FIG. 4C

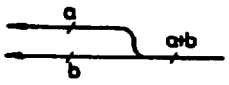


FIG. 4D

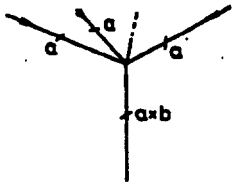


FIG. 4E

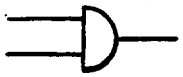


FIG. 4F

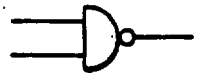


FIG. 4G

FIG. 4H

FIG. 4I

FIG. 4J

FIG. 4K

FIG. 4L

FIG. 4M

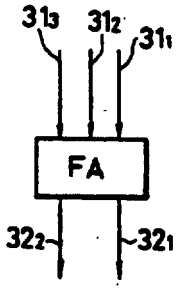


FIG. 4N

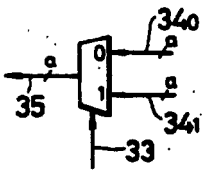


FIG. 4P

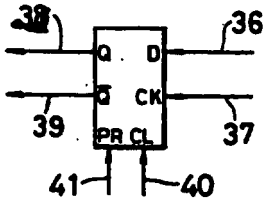


FIG. 4Q

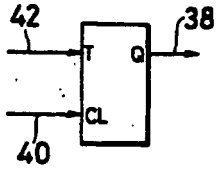


FIG. 4R

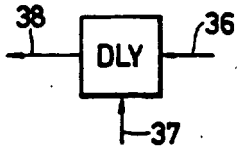


FIG. 4S

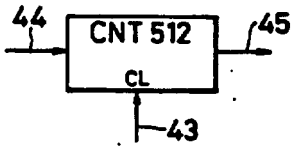


FIG. 4T

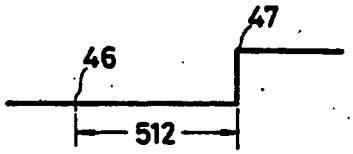


FIG. 4U

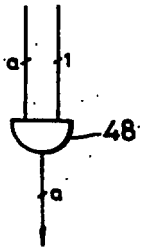


FIG. 4V

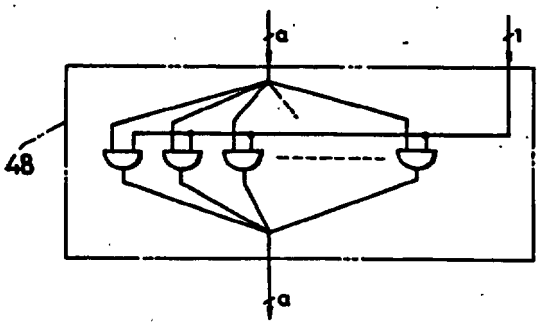


FIG. 4W

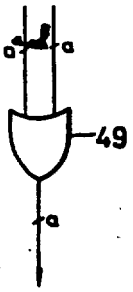


FIG. 4X

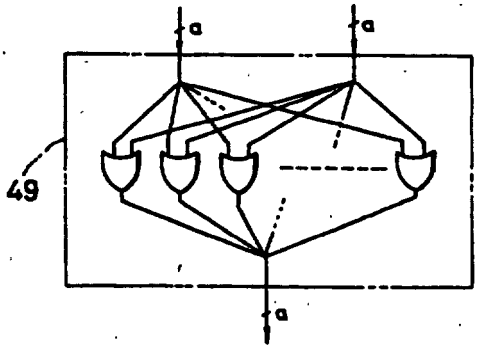


FIG. 4Y

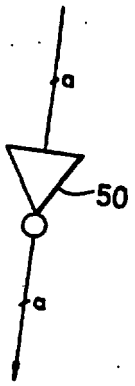


FIG. 4Z

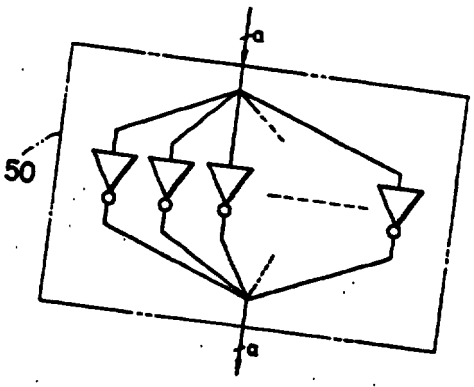


FIG. 5A

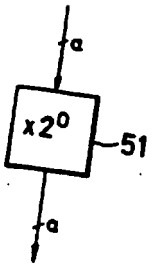


FIG. 5B

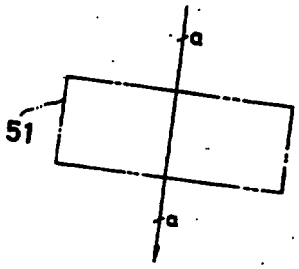


FIG. 5C

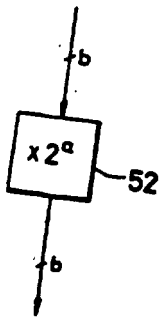


FIG. 5D

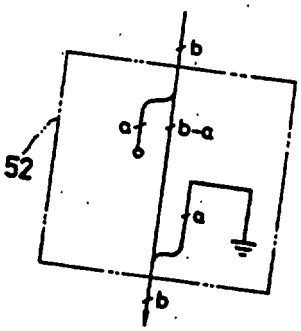


FIG. 5E

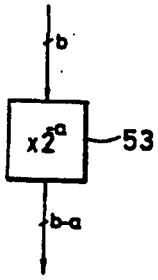


FIG. 5F

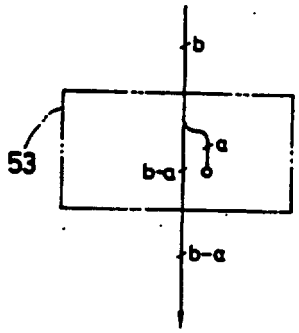


FIG. 5G

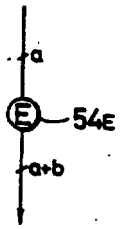


FIG. 5H

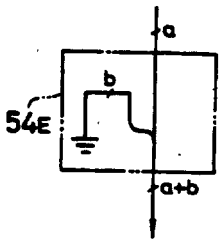


FIG. 5I

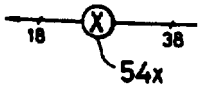


FIG. 5J

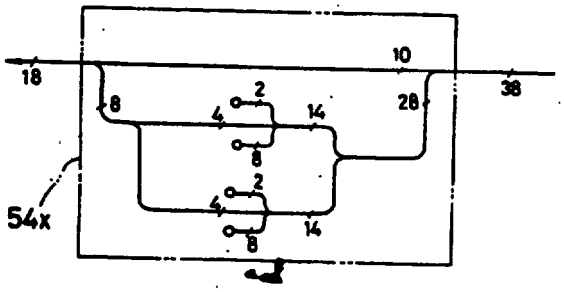


FIG. 5K

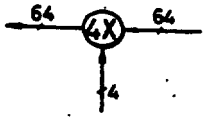


FIG. 5L

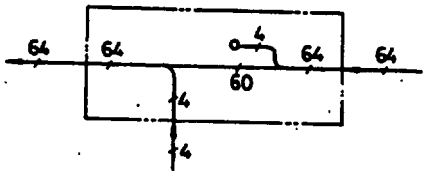


FIG. 5M

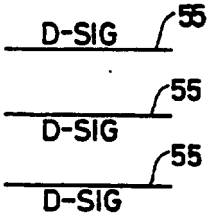


FIG. 5N

FIG. 5P

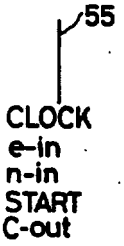


FIG. 5Q

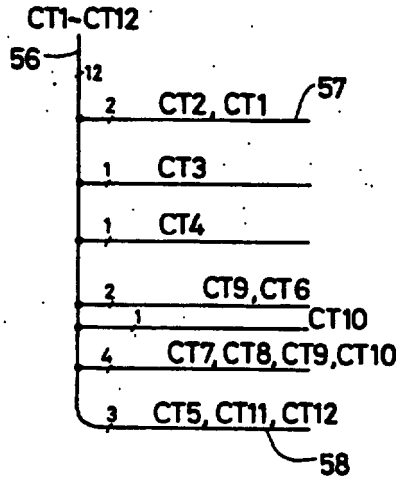
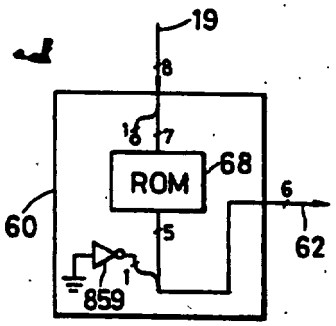


FIG. 7



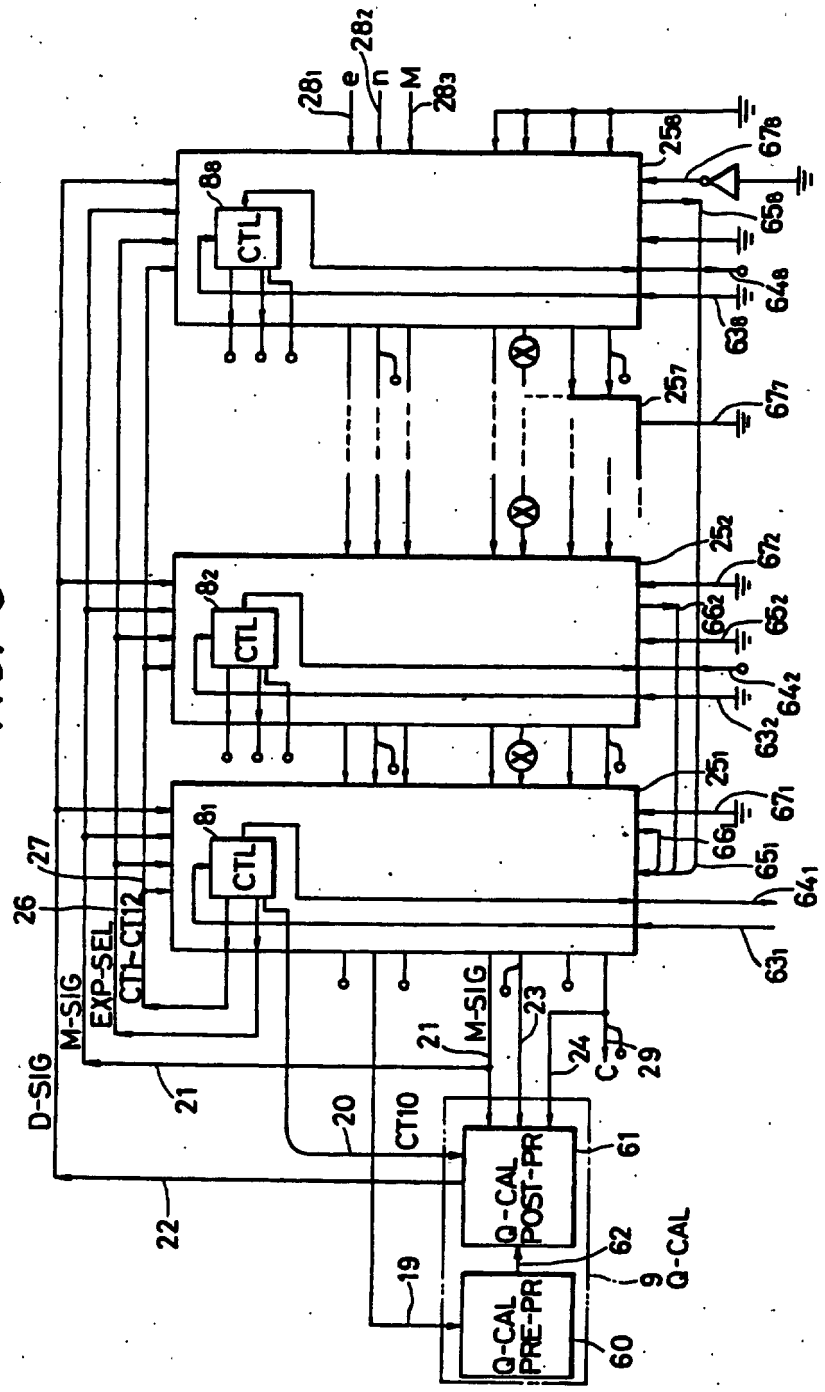

281
e
282
n
M
283
CTL
CTL
CTL
M-SIG
EXP-SEL
CTI-CTI2
D-SIG
M-SIG
M-SIG
CT10
Q-CAL
POST-PR
Q-CAL
PRE-PR
Q-CAL

FIG. 6

FIG. 8

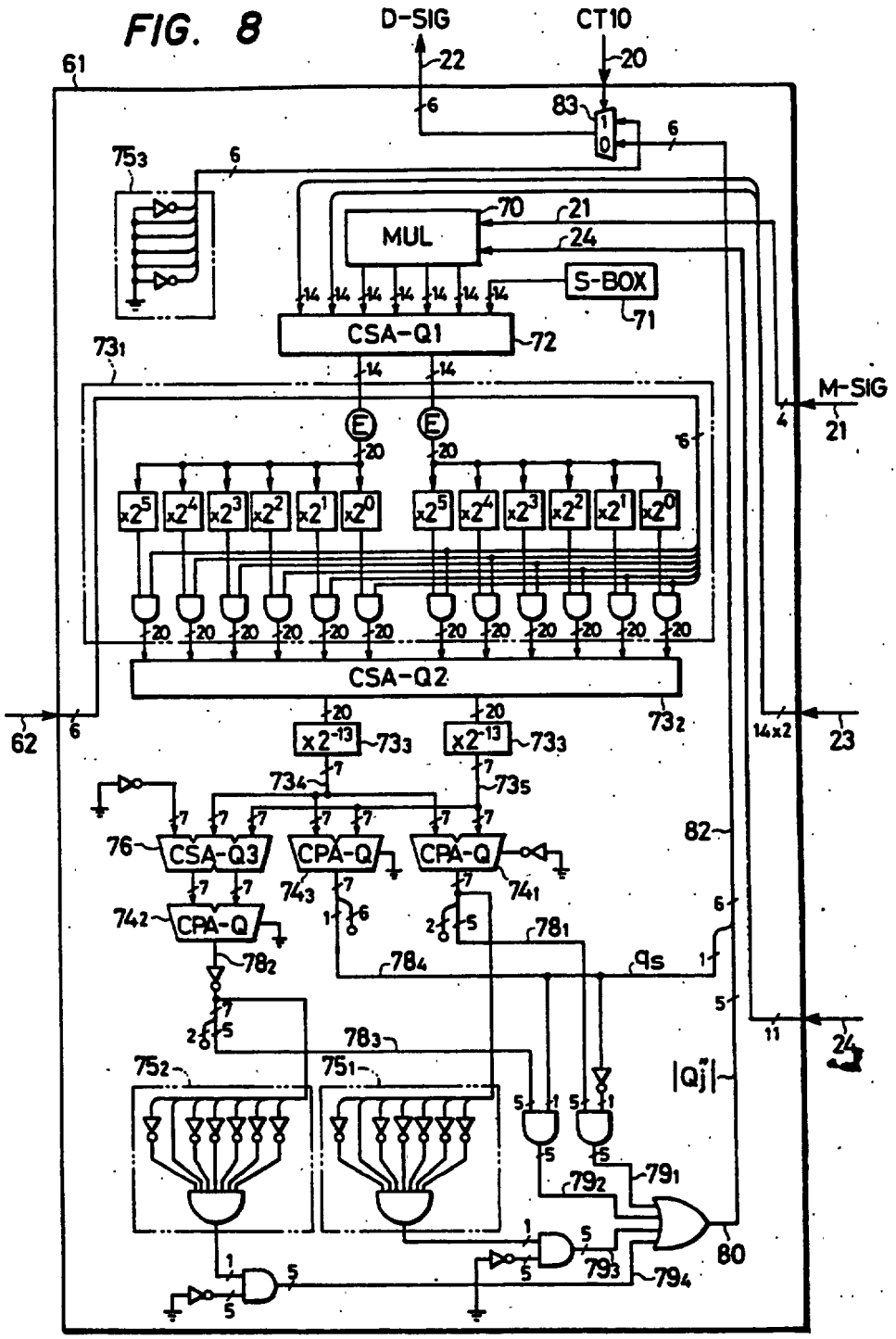


FIG. 9

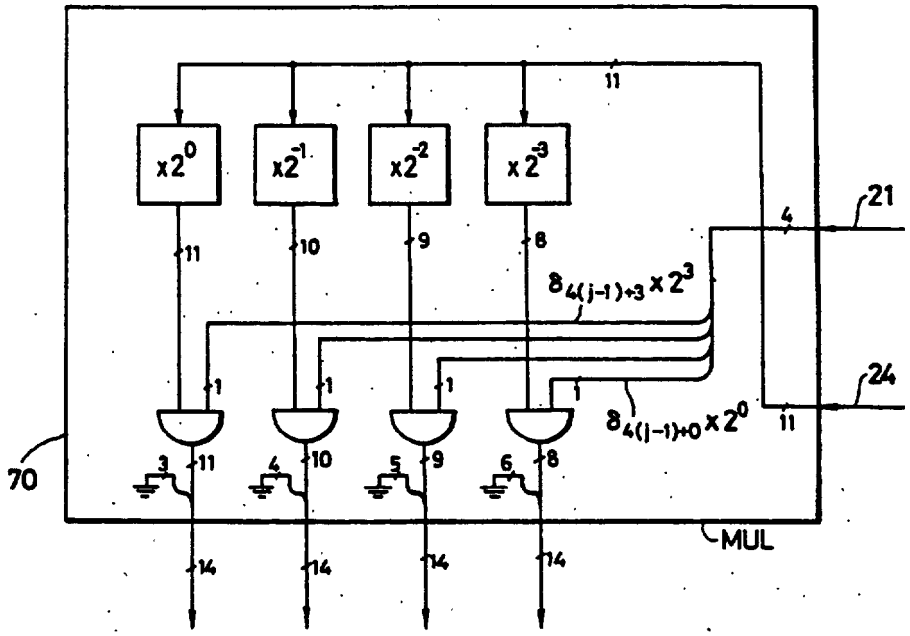

x2^0
x2^-1
x2^-2
x2^-3
11
10
9
8
4
21
24
δ4(j-1)+3 x 2^3
δ4(j-1)+0 x 2^0
70
MUL
14

FIG. 10

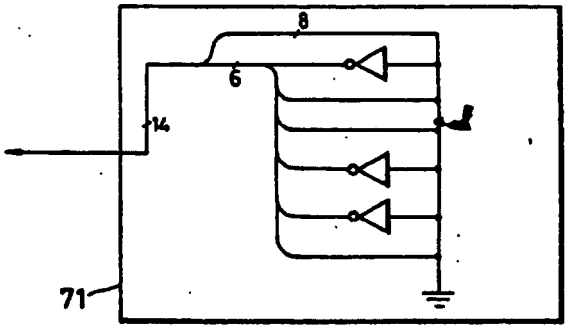

8
6
14
71

FIG. 11

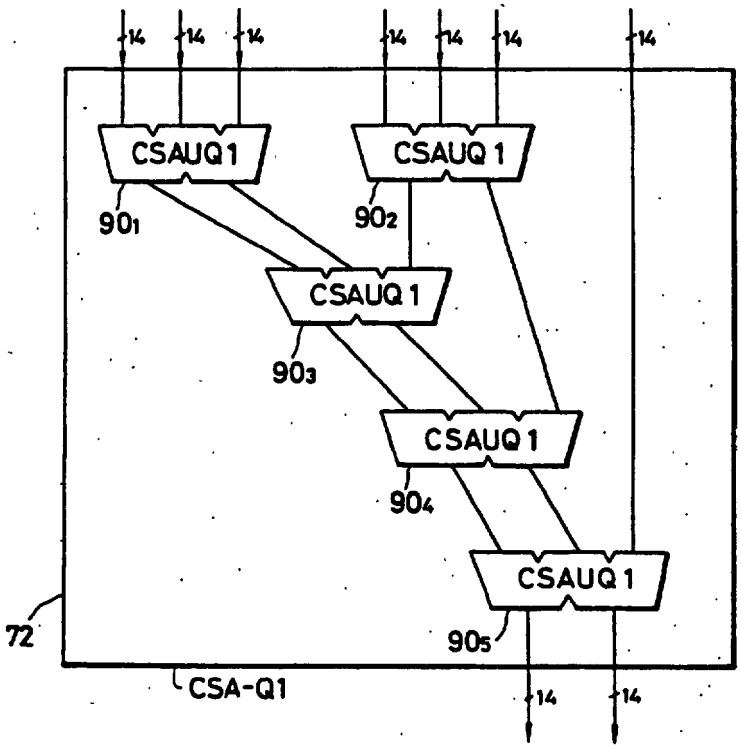


FIG. 12

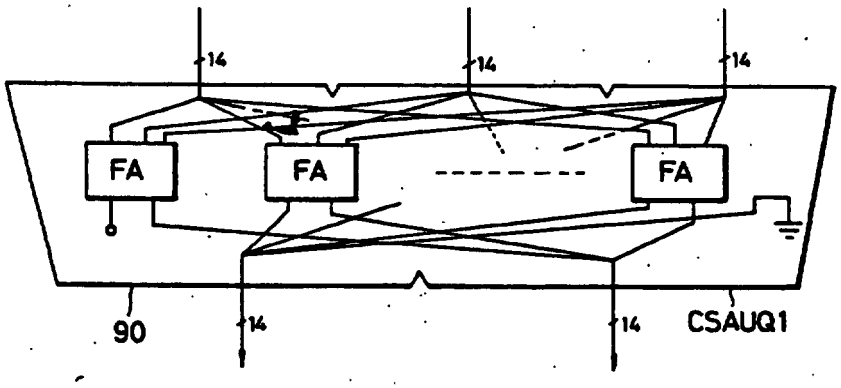


FIG. 13

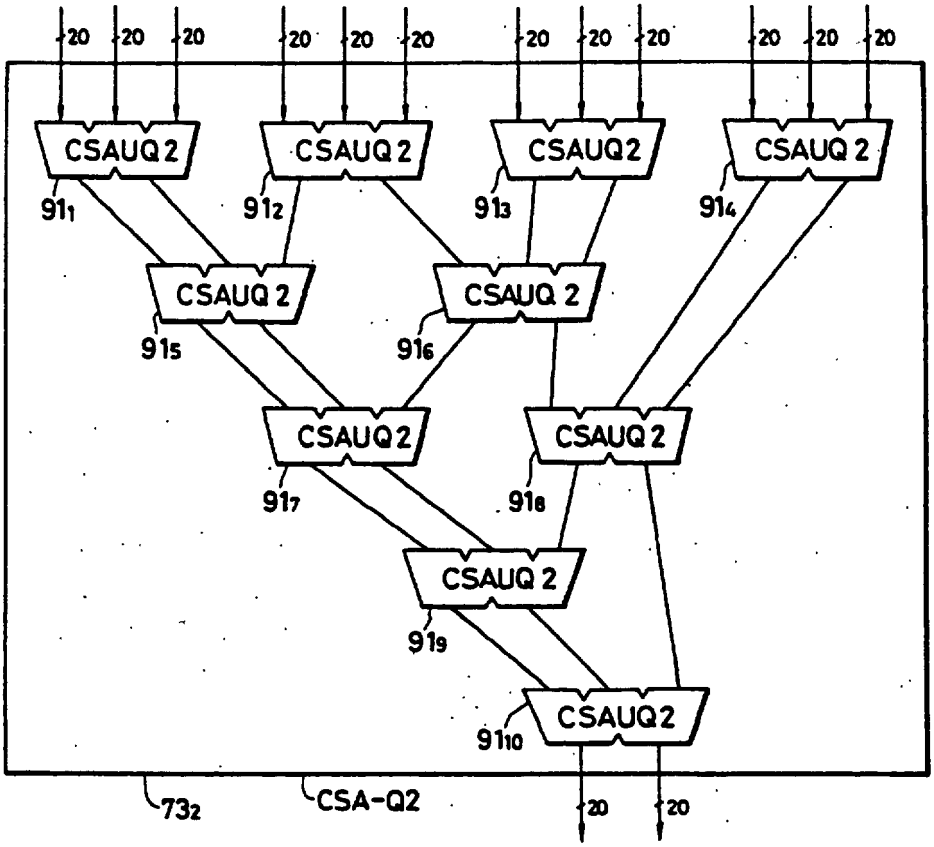

20
CSAUQ2
91₁
91₂
91₃
91₄
91₅
91₆
91₇
91₈
91₉
91₁₀
73₂
CSA-Q2

FIG. 14

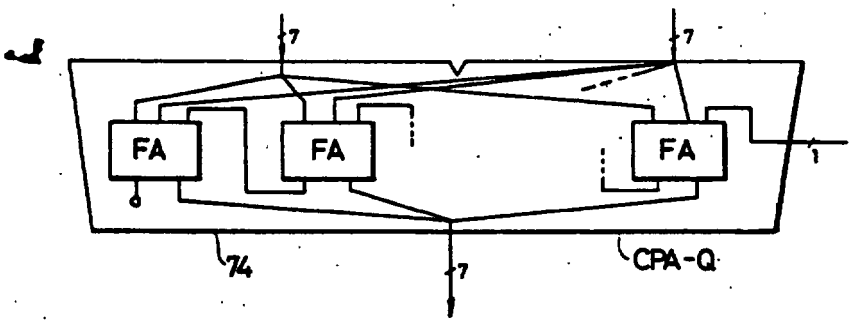
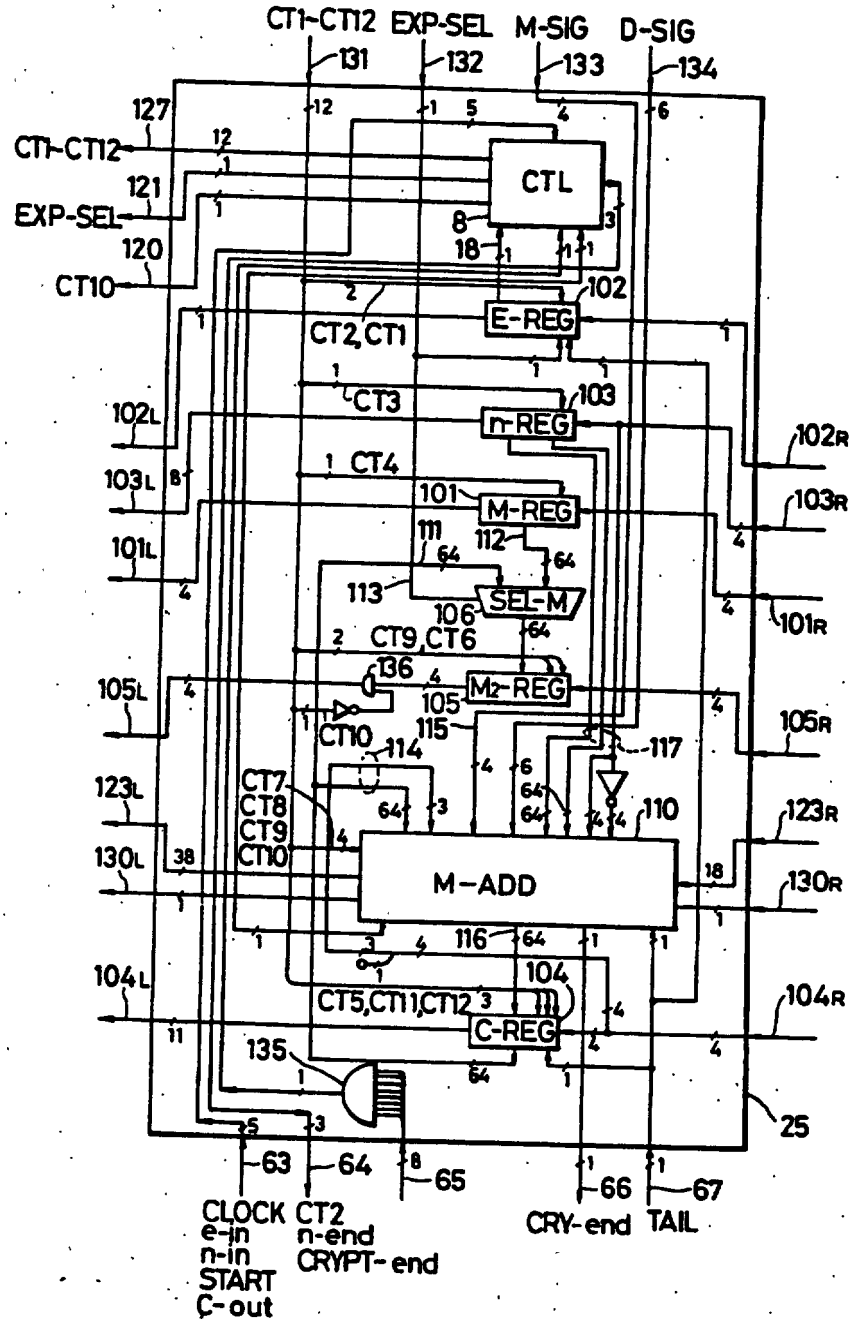

7
FA
0
1
74
CPA-Q

FIG. 15


CT1-CT12
EXP-SEL
M-SIG
D-SIG
131
132
133
134
127
CT1-CT12
121
EXP-SEL
120
CT10
CTL
E-REG
CT2,CT1
CT3
n-REG
103
102L
102R
CT4
101
M-REG
103L
103R
101L
111
112
113
SEL-M
106
101R
CT9,CT6
136
M2-REG
105L
105
105R
CT10
114
115
117
CT7
CT8
CT9
CT10
123L
123R
110
130L
M-ADD
130R
116
104
104L
CT5,CT11,CT12
C-REG
104R
135
25
63
64
65
66
67
CLOCK
e-in
n-in
START
C-out
CT2
n-end
CRYPT-end
CRY-end
TAIL

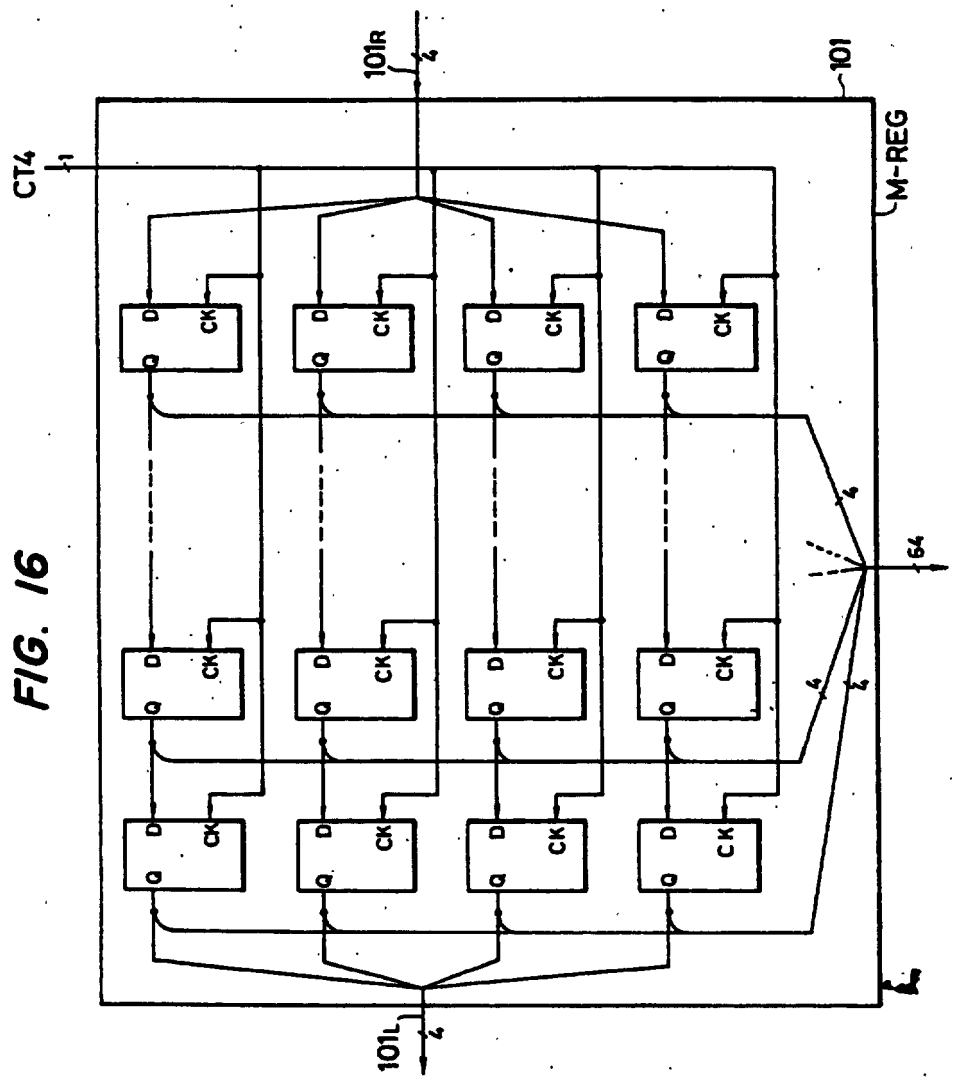


FIG. 16

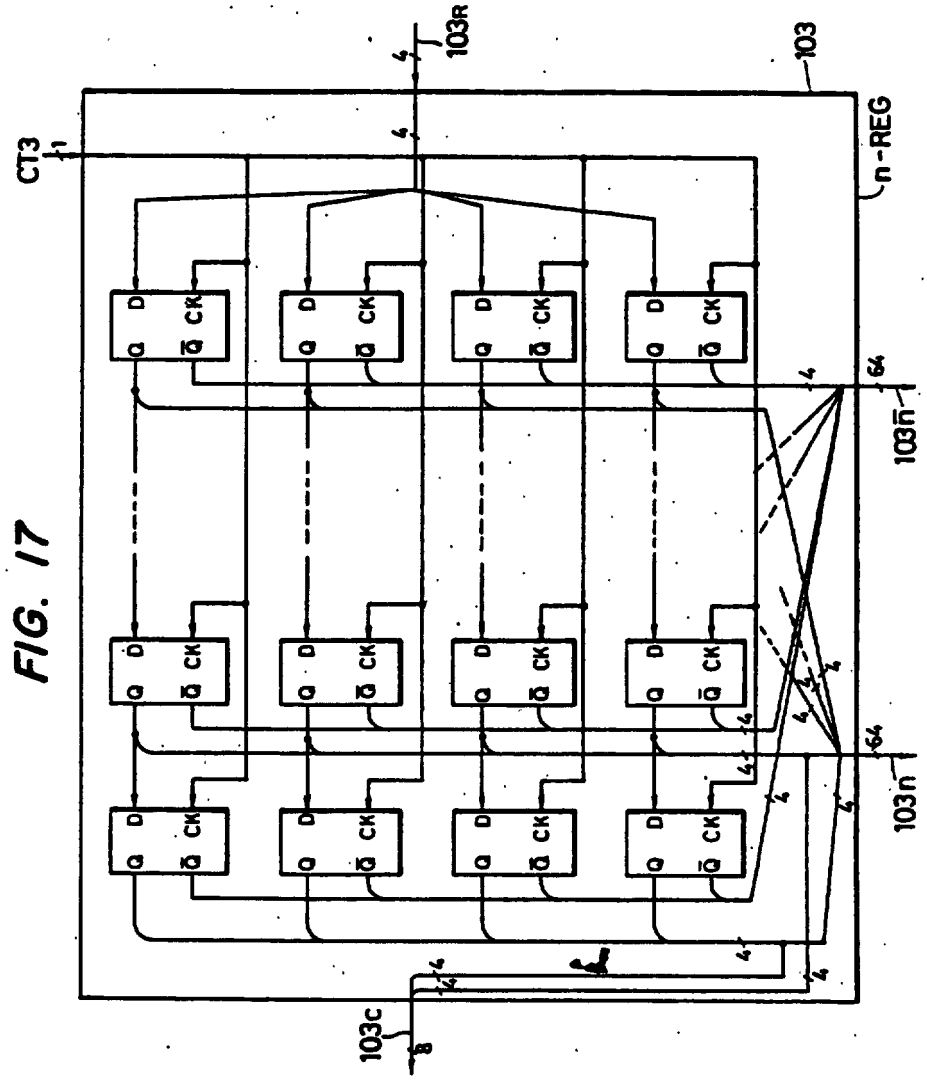
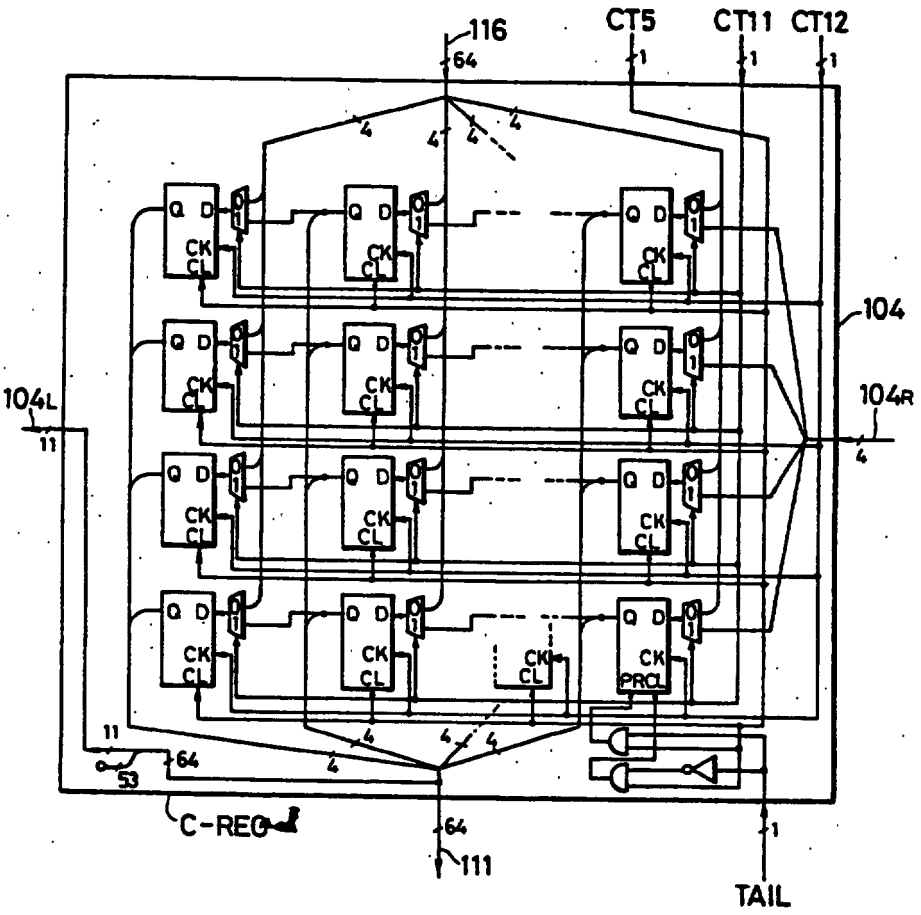


FIG. 17

FIG. 18


116
64
CT5
1
CT11
1
CT12
1
104
104R
4
104L
11
Q D
CK
CL
PRCL
11
64
53
C-REG
64
111
1
TAIL

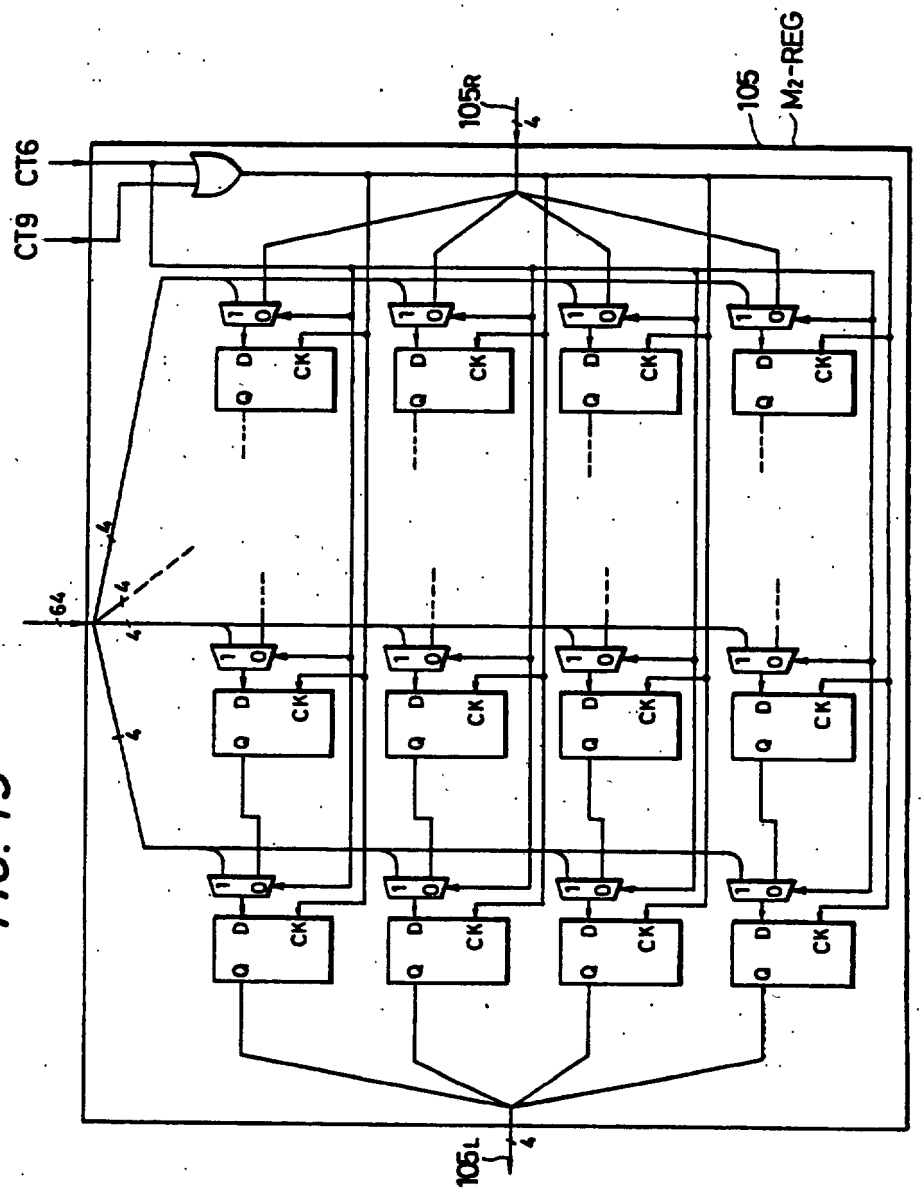


FIG. 19

FIG. 20

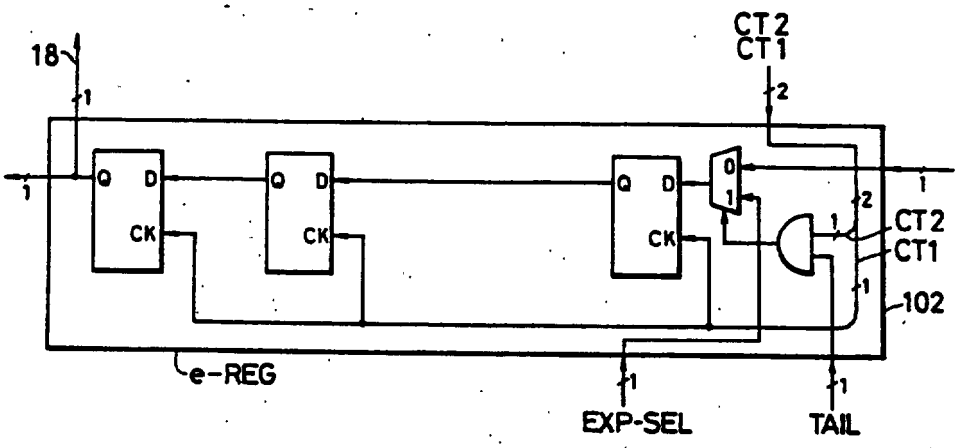
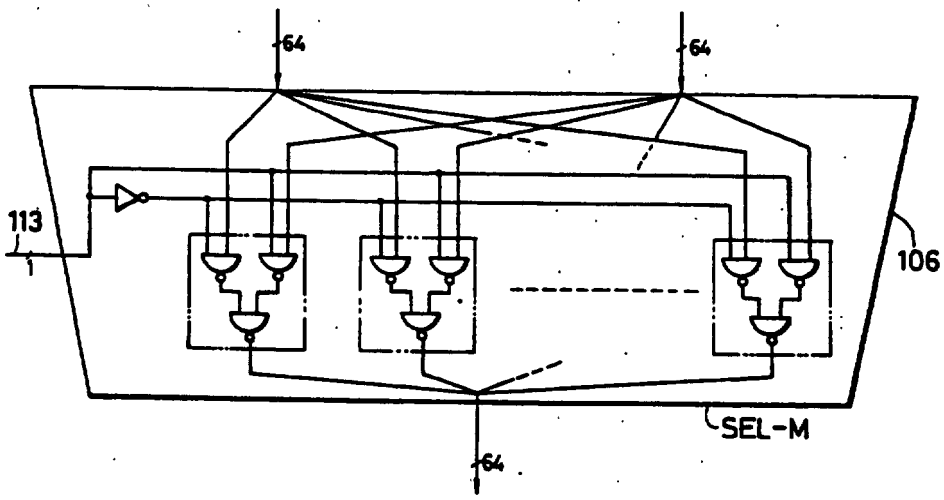

18
1
CT2
CT1
2
Q D
CK
1
2
CT2
CT1
102
e-REG
EXP-SEL
TAIL

FIG. 21


64
64
113
106
SEL-M
64

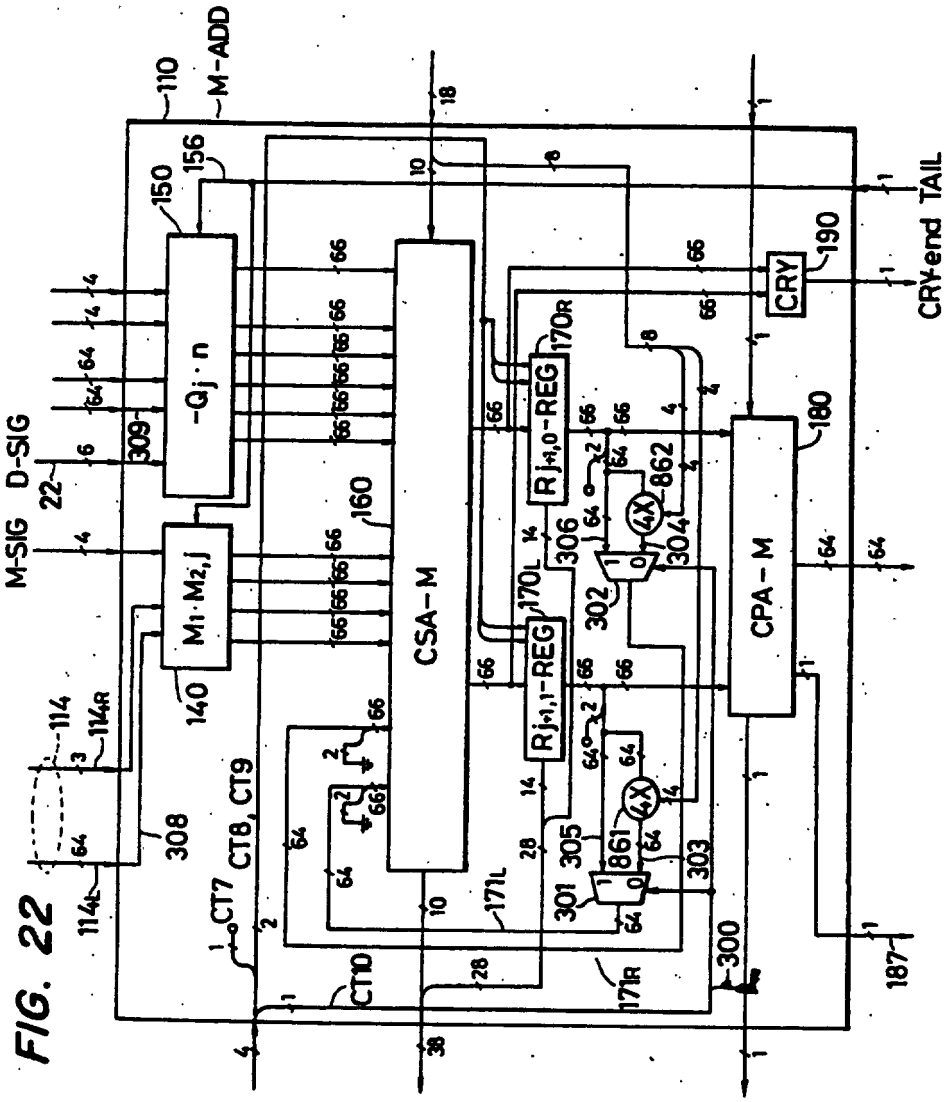
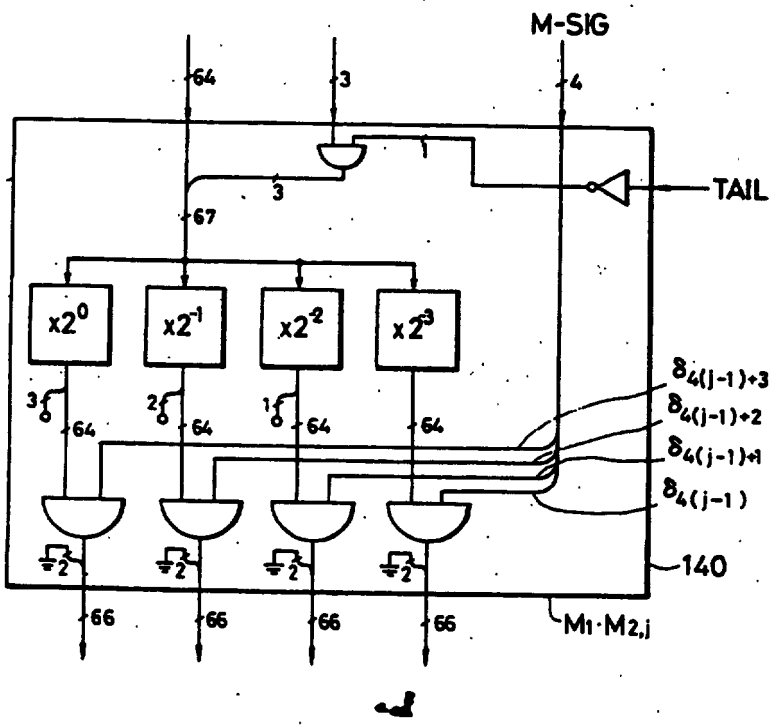

M-ADD
110
150
156
-Qj · n
M1 · M2,j
M-SIG
D-SIG
22
309
140
160
CSA-M
CRY
190
CRY-end TAIL
180
CPA-M
Rj+1,0-REG
Rj+1,1-REG
170R
170L
171L
171R
862
861
4X
306
304
302
305
303
301
300
187
308
114
114R
CT7
CT8, CT9
CT10

FIG. 22

FIG. 23


M-SIG
64
3
4
3
TAIL
67
x2^0
x2^-1
x2^-2
x2^-3
3
2
1
64
64
64
64
δ4(j-1)+3
δ4(j-1)+2
δ4(j-1)+1
δ4(j-1)
140
÷2
66
66
66
66
M1·M2,j

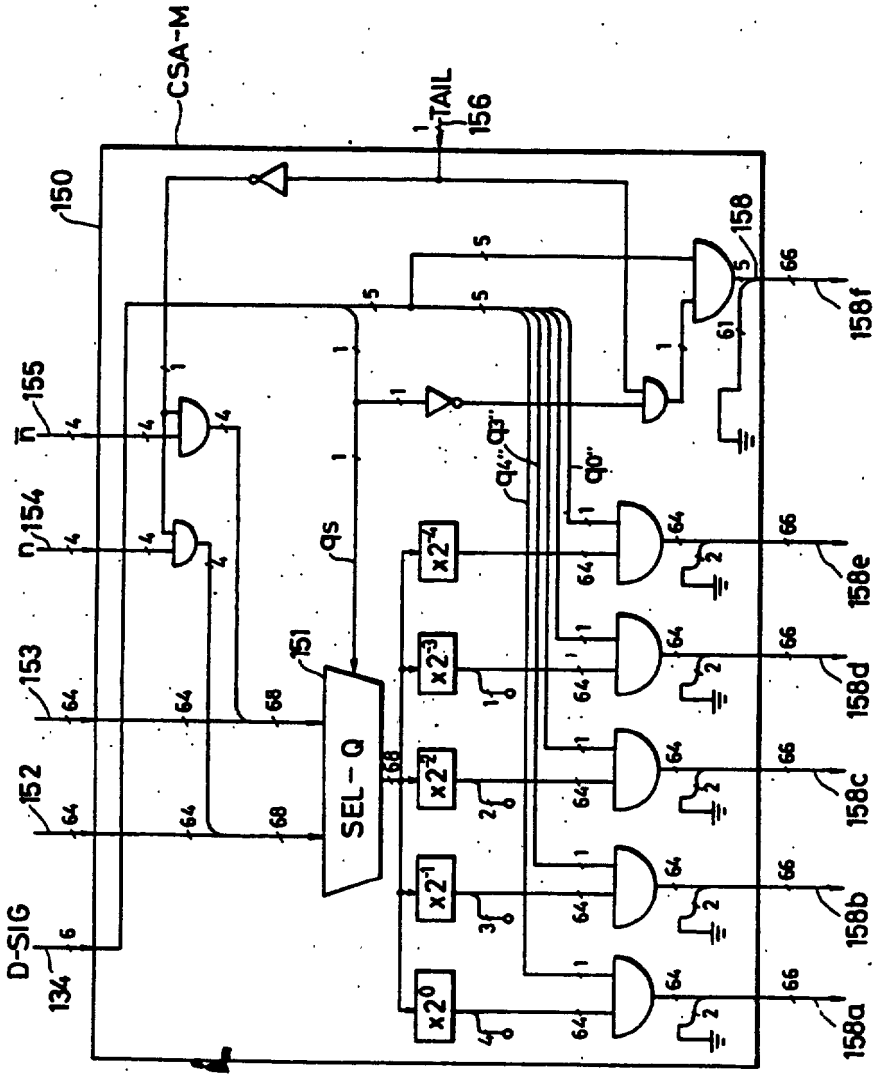

CSA-M
TAIL
156
150
158
158f
155
154
153
152
151
SEL - Q
D-SIG
134
158e
158d
158c
158b
158a

FIG. 24

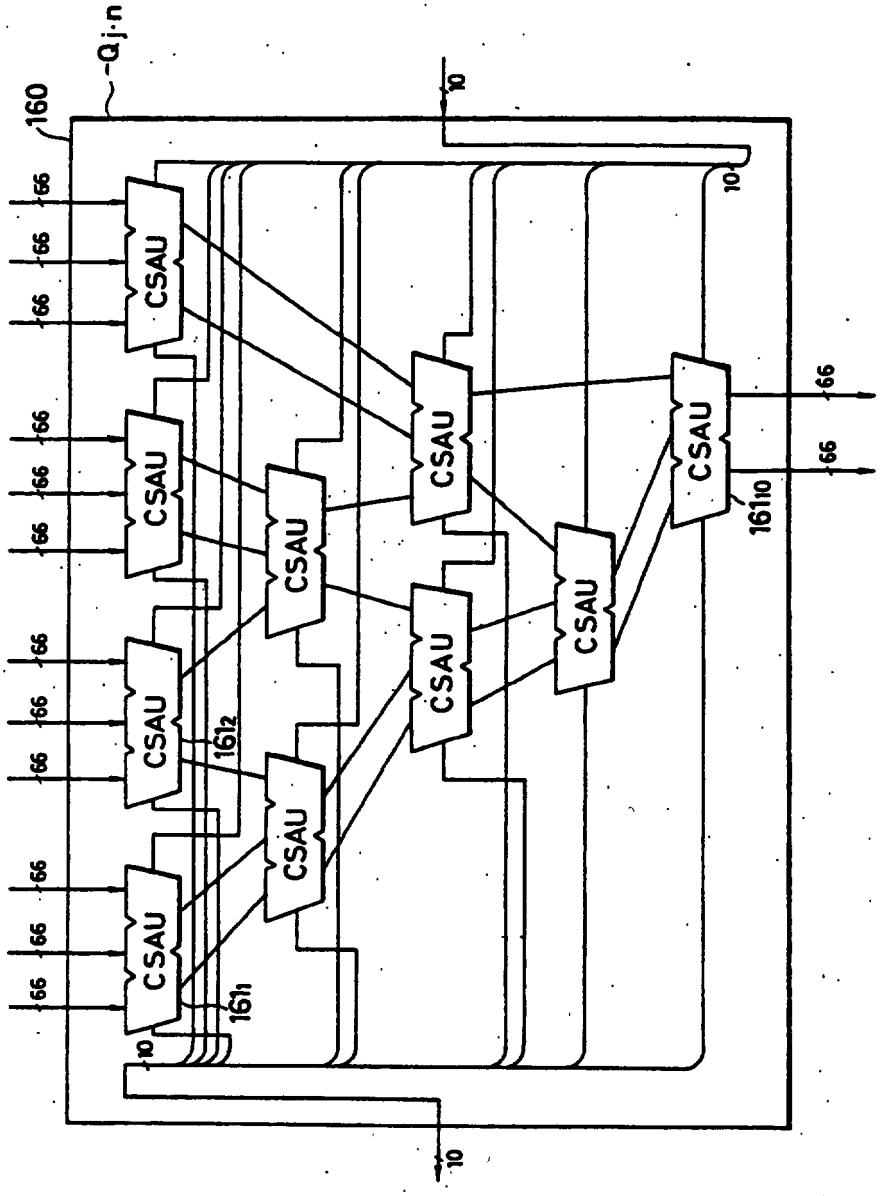


FIG. 25

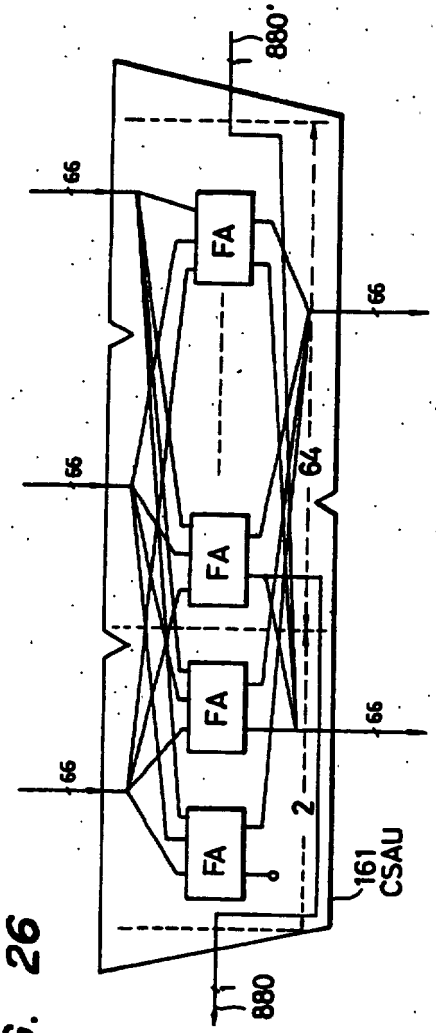

880'
66
FA
66
66
64
FA
66
FA
66
2
FA
161
CSAU
880

FIG. 26

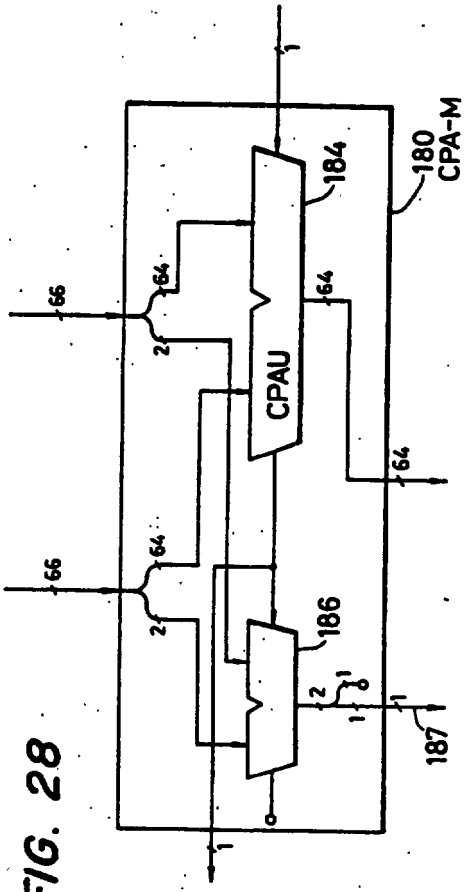

1
180
CPA-M
184
66
64
2
CPAU
64
64
66
64
2
186
2
1
1
1
187
1

FIG. 28

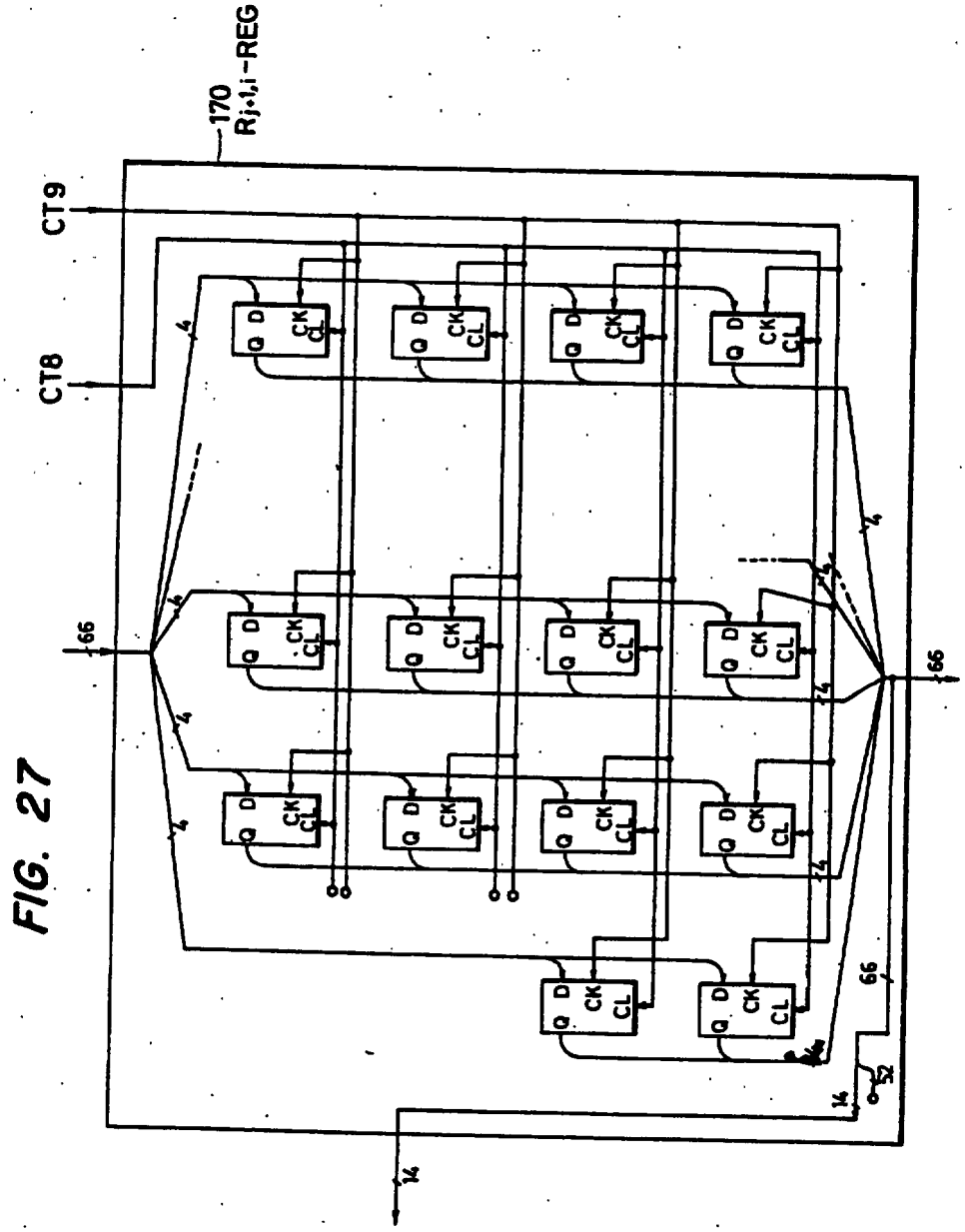


FIG. 27

FIG. 29

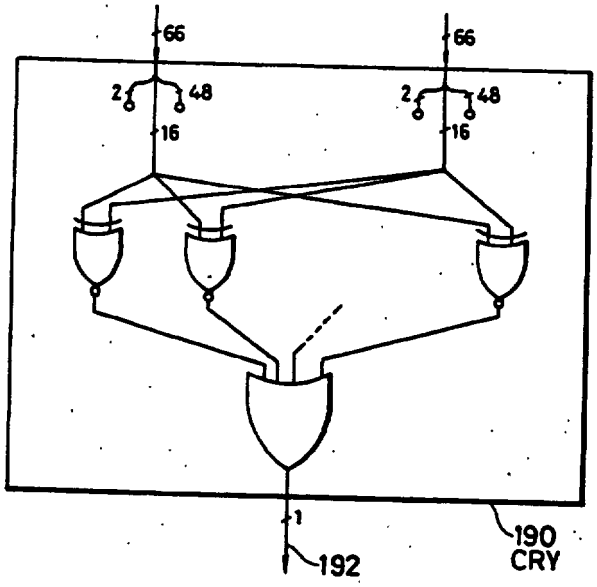

66
66
2
48
16
2
48
16
1
192
190
CRY

FIG. 30

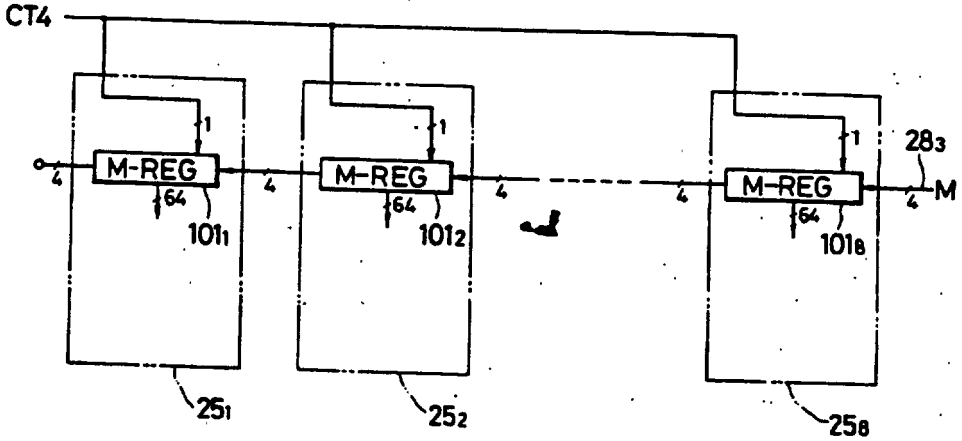

CT4
1
M-REG
64
101₁
4
1
M-REG
64
101₂
4
4
1
M-REG
64
101₈
28₃
4
M
25₁
25₂
25₈

FIG. 31

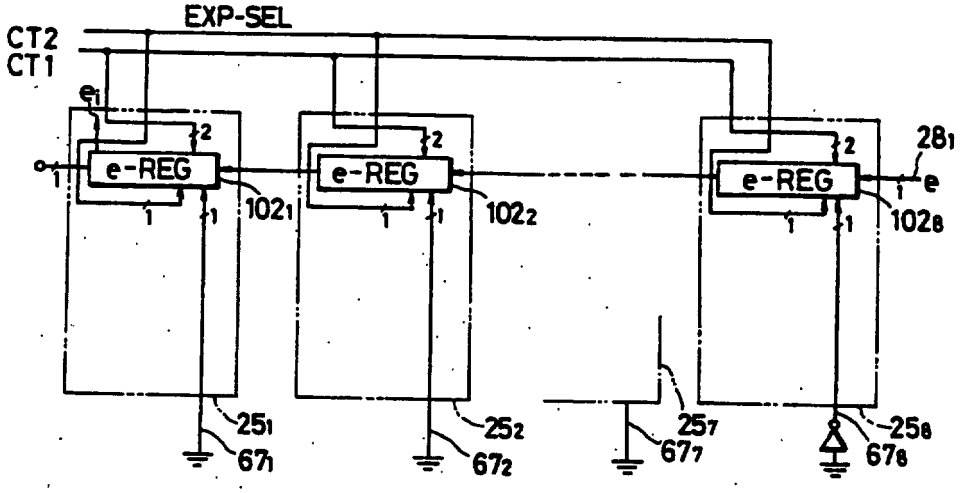

EXP-SEL
CT2
CT1
e-REG
e-REG
e-REG
10_{2_1}
10_{2_2}
10_{2_8}
28_1
e
25_1
25_2
25_7
25_8
67_1
67_2
67_7
67_8

FIG. 32

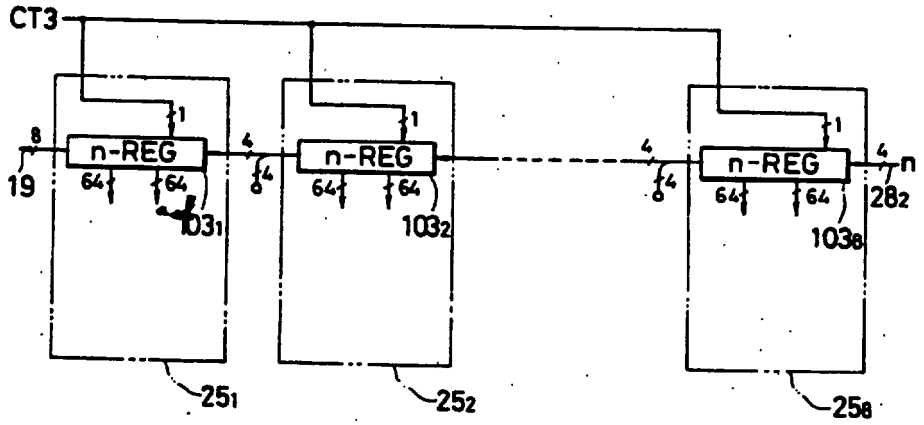

CT3
n-REG
n-REG
n-REG
19
10_{3_1}
10_{3_2}
10_{3_8}
28_2
n
25_1
25_2
25_8

FIG. 33

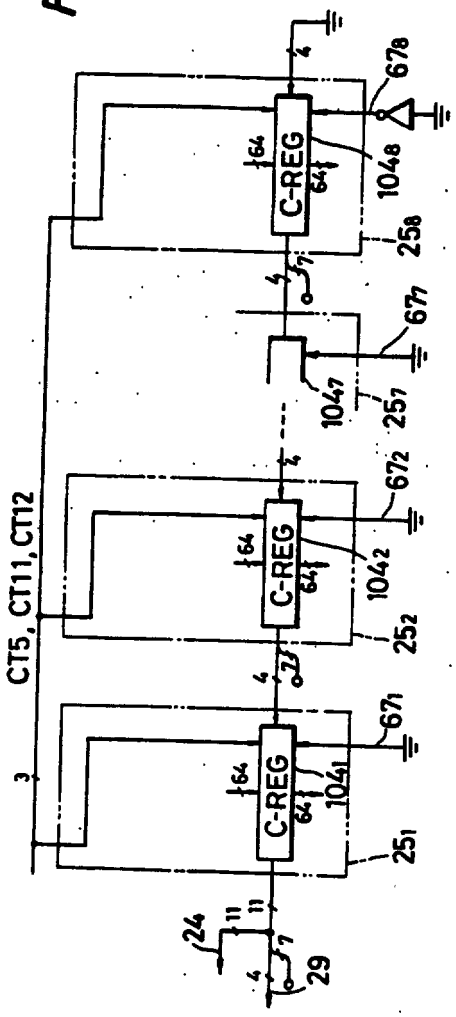

C-REG
64
64
1048
1042
1041
1047
678
677
672
671
258
257
252
251
CT5, CT11, CT12
24
29
11
7
4
3

FIG. 34

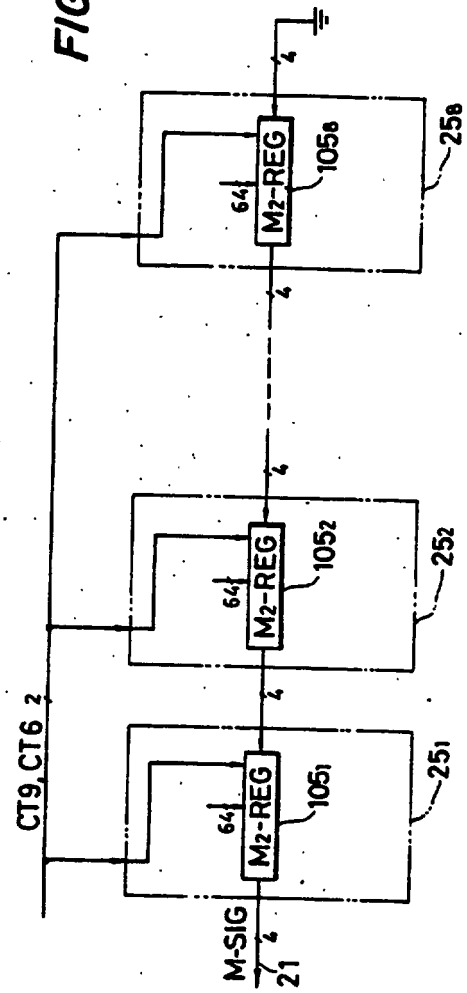

M2-REG
64
1058
1052
1051
258
252
251
CT9, CT6
M-SIG
21
4
2

FIG. 35

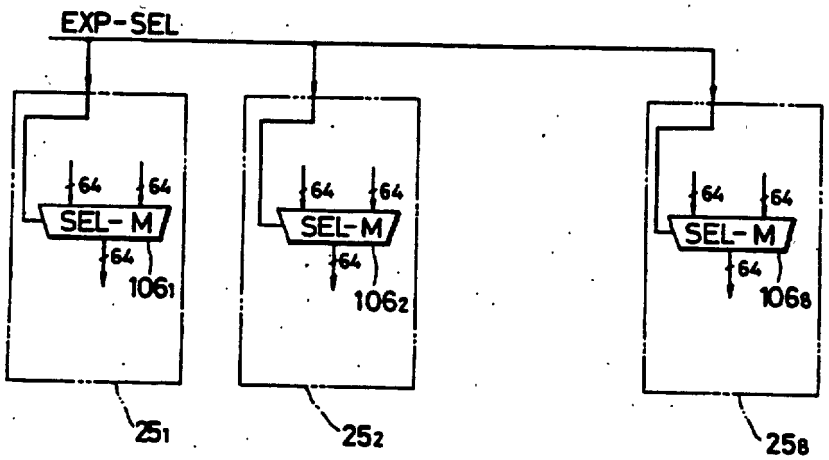

EXP-SEL
64
64
SEL-M
64
106₁
106₂
106₈
25₁
25₂
25₈

FIG. 36

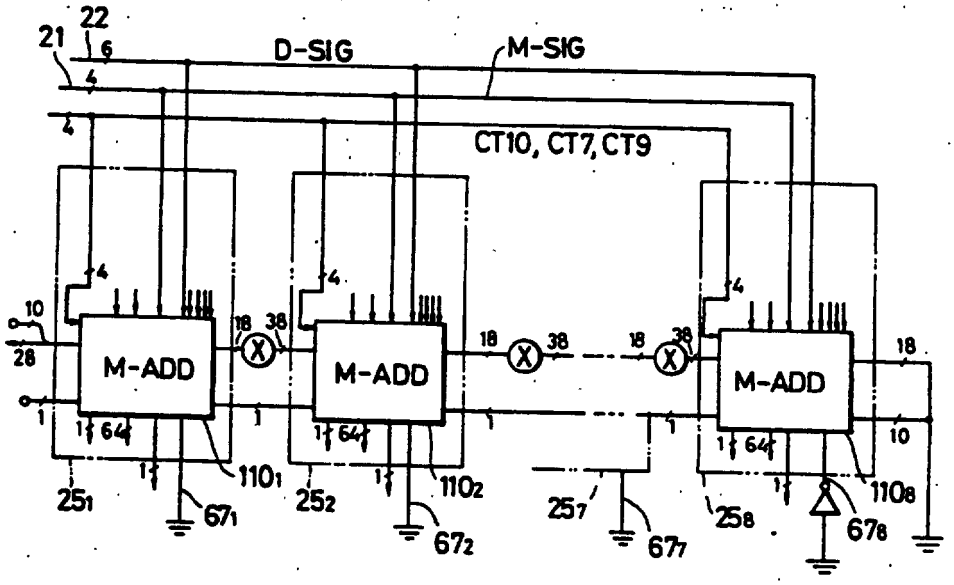

21
22
6
4
D-SIG
M-SIG
CT10, CT7, CT9
10
28
M-ADD
18
38
64
110₁
110₂
110₈
25₁
25₂
25₇
25₈
67₁
67₂
67₇
67₈

FIG. 37

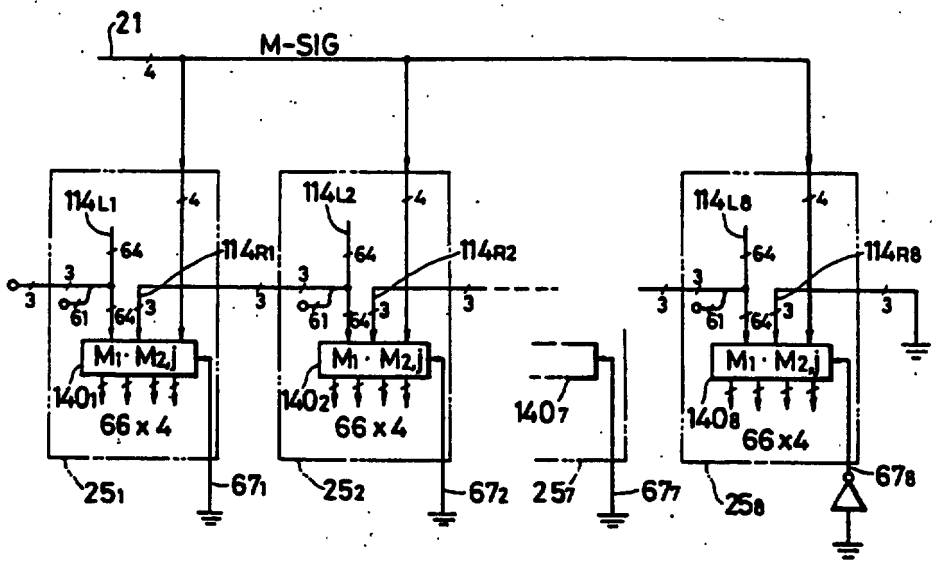

21
M-SIG
4
114L1
114L2
114L8
64
114R1
114R2
114R8
3
61
M1·M2j
1401
1402
1407
1408
66x4
251
252
257
258
671
672
677
678

FIG. 38

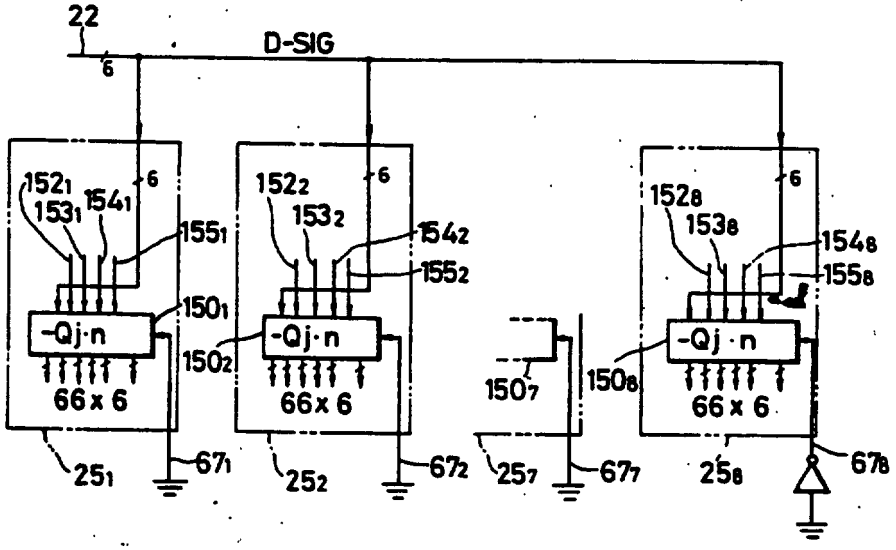

22
D-SIG
6
1521
1531
1541
1551
1522
1532
1542
1552
1528
1538
1548
1558
-Qj·n
1501
1502
1507
1508
66x6
251
252
257
258
671
672
677
678

FIG. 39

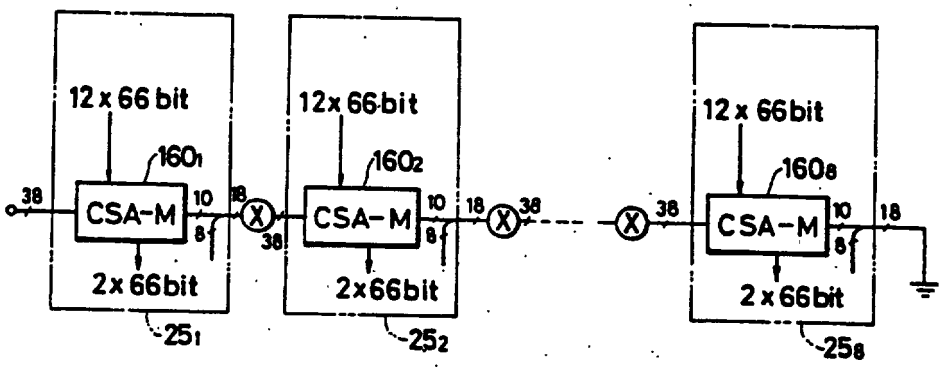

12 x 66 bit
160₁
CSA-M
2x66bit
25₁
12x66bit
160₂
CSA-M
2x66bit
25₂
12 x 66bit
160₈
CSA-M
2x66bit
25₈
38
10
18
8
X

FIG. 40

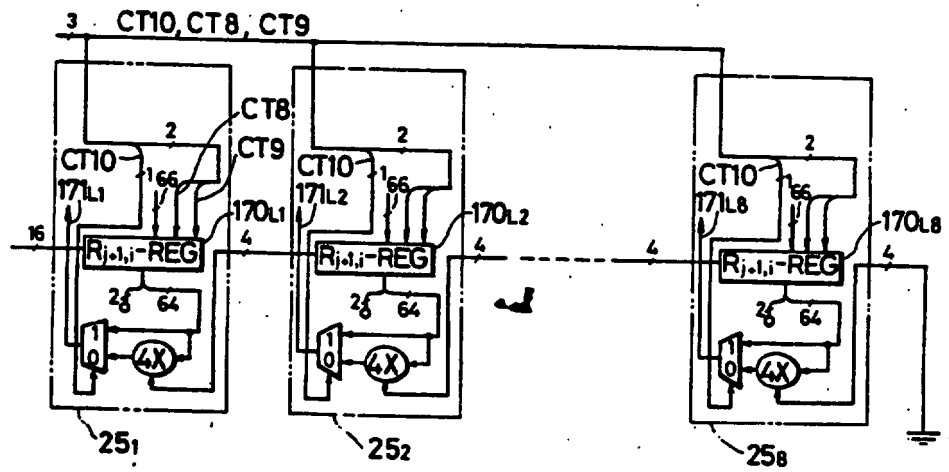

3
CT10, CT8, CT9
CT8
CT9
CT10
171L1
171L2
171L8
2
166
16
4
Rj-1,i-REG
170L1
170L2
170L8
2
64
4X
25₁
25₂
25₈

FIG. 41

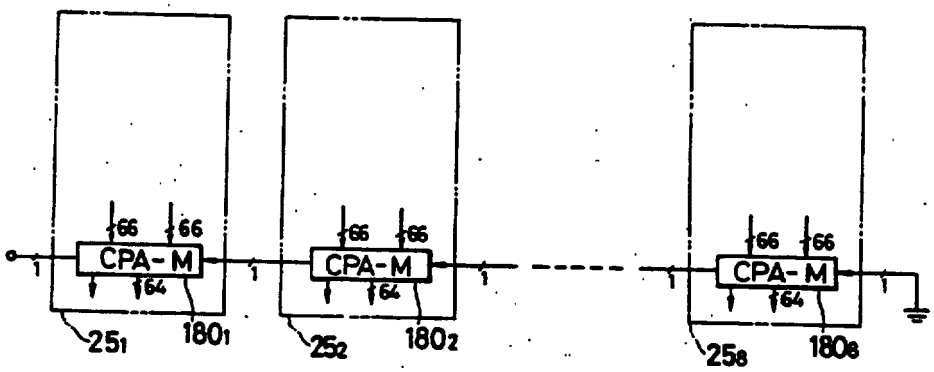

66
66
CPA-M
64
251
1801
252
1802
258
1808

FIG. 42

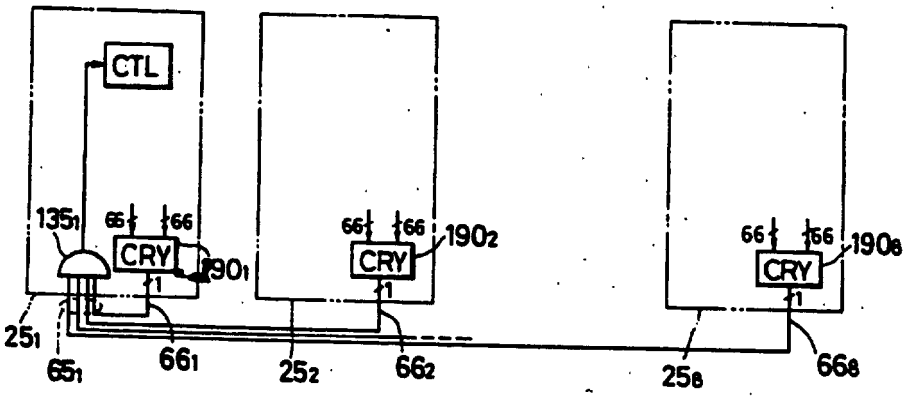

CTL
1351
65
66
CRY
1901
1902
1908
251
651
661
252
662
258
668

FIG. 43

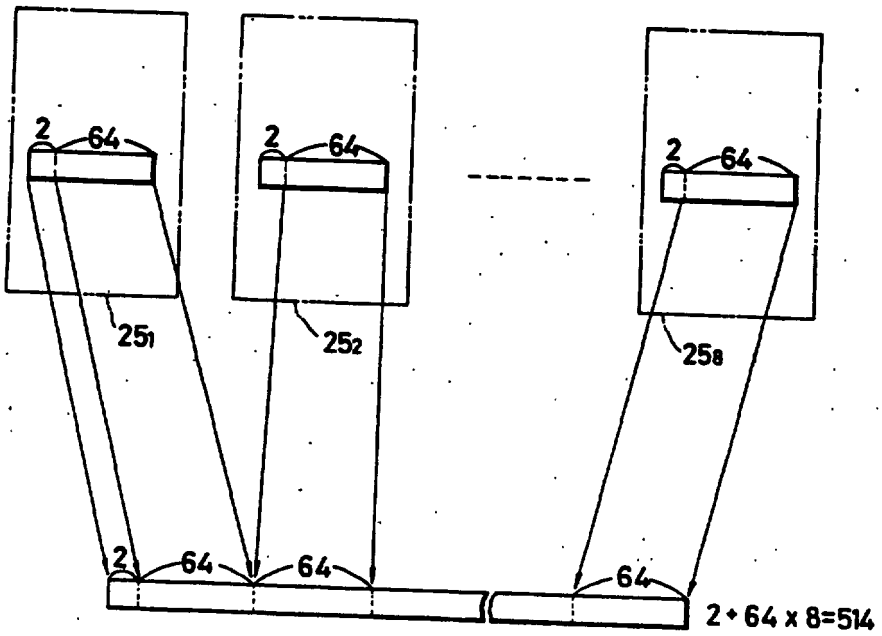


FIG. 44

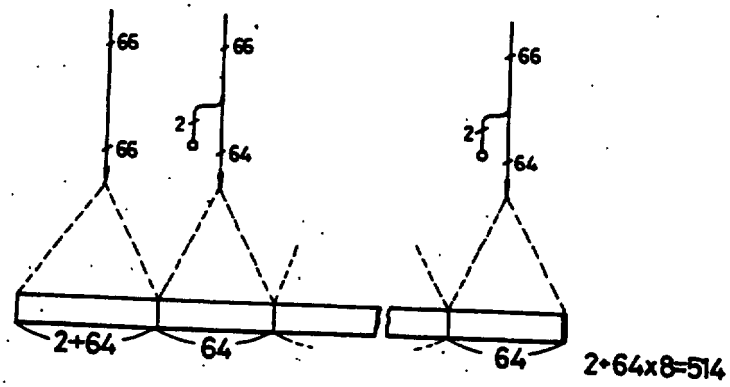

66
66
2
64
66
2
64
66
2
64
2+64
64
64
2+64x8=514

FIG. 45

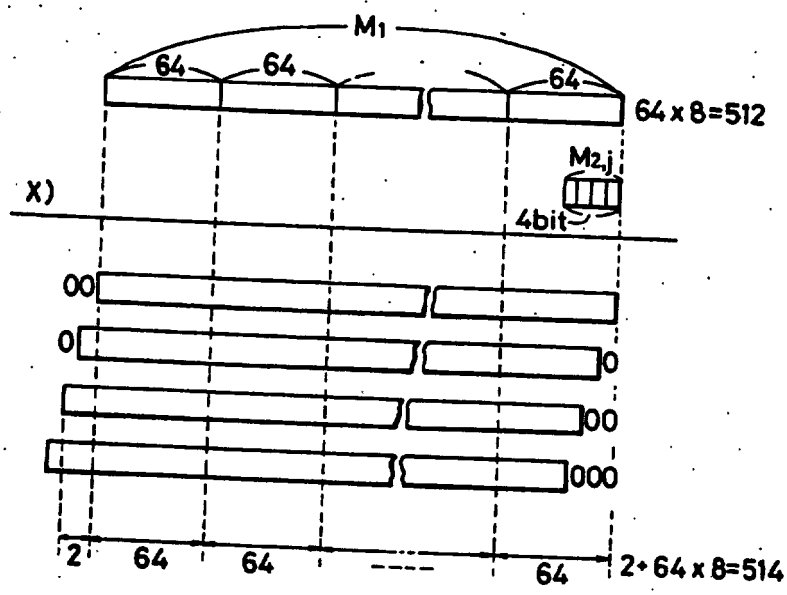

M1
64
64
64
64 x 8=512
M2,j
X)
4bit
00
0
0
00
000
2
64
64
64
2+64 x 8=514

FIG. 46

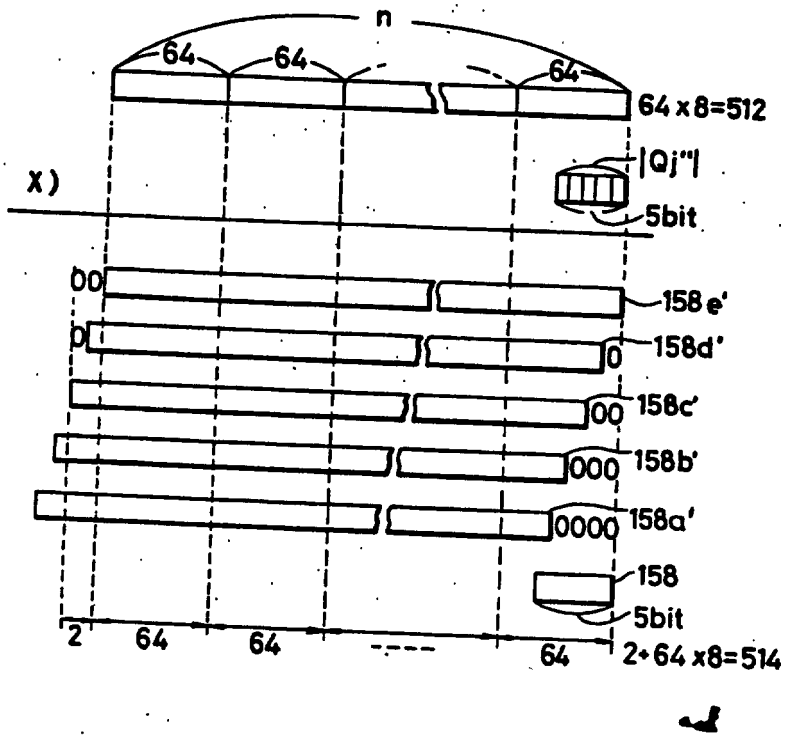

n
64
64
64
64 x 8=512
|Qj''|
5bit
X)
00
158 e'
0
158d'
00
158c'
000
158b'
0000
158a'
158
5bit
2
64
64
64
2+64 x8=514

FIG. 47

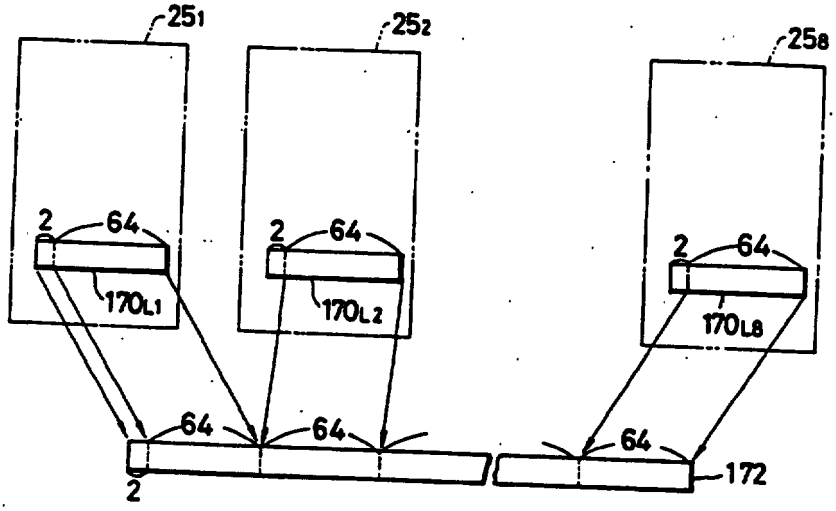
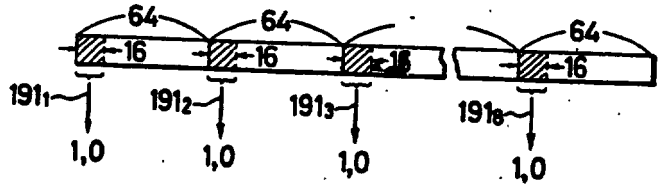

251
252
258
2
64
170L1
170L2
170L8
64
64
64
2
172

FIG. 48


64
64
64
16
16
16
191₁
191₂
191₃
191₈
1,0
1,0
1,0
1,0

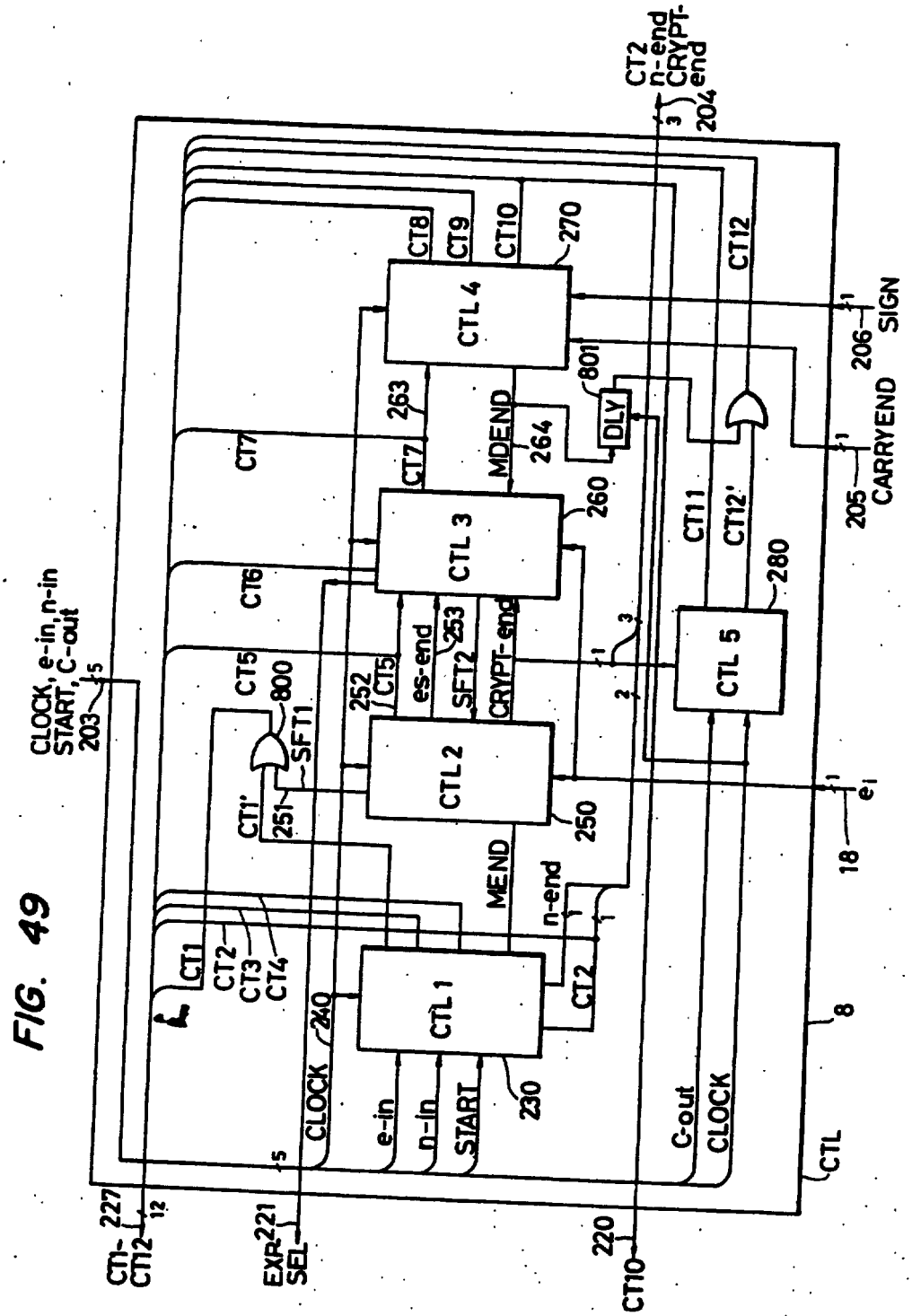

CT2
n-end
CRYPT-end
204
CT8
CT9
CT10
270
CT12
CTL 4
801
DLY
206
SIGN
263
CT7
MDEND
264
CT7
260
CT11
CT12'
280
205
CARRYEND
CT6
CTL 3
CTL 5
CLOCK, e-in, n-in
START, C-out
203
CT5
800
SFT1
252
CT5
es-end
253
SFT2
CRYPT-end
CT1'
251
CTL 2
250
e1
18
MEND
n-end
CT1
CT2
CT3
CT4
CTL 1
CT2
240
CLOCK
e-in
n-in
START
230
C-out
CLOCK
8
CTL
227
12
CT1-
CT12
221
5
EXR
SEL
220
CT10

FIG. 49

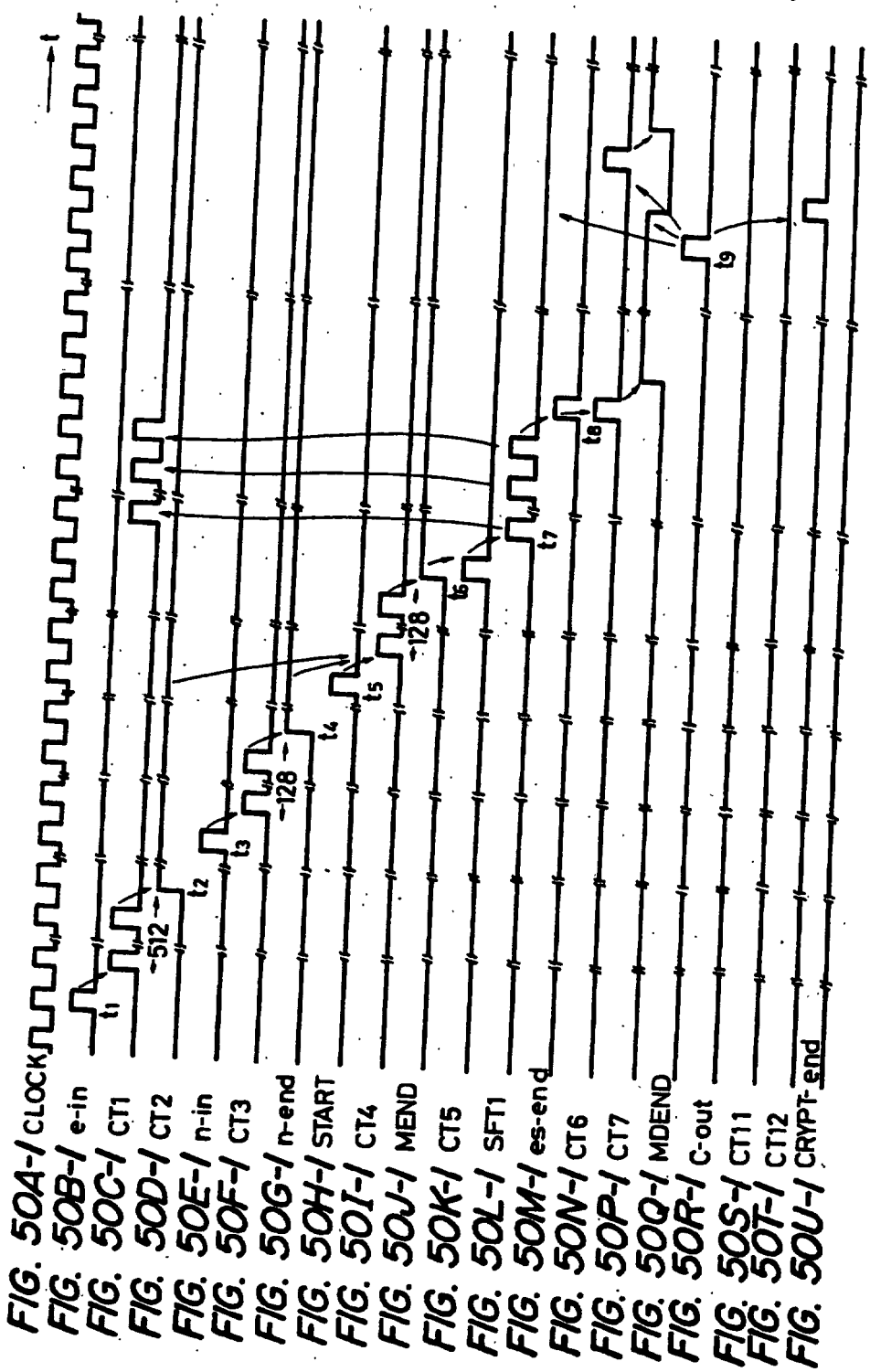


FIG. 50A-1
FIG. 50B-1
FIG. 50C-1
FIG. 50D-1
FIG. 50E-1
FIG. 50F-1
FIG. 50G-1
FIG. 50H-1
FIG. 50I-1
FIG. 50J-1
FIG. 50K-1
FIG. 50L-1
FIG. 50M-1
FIG. 50N-1
FIG. 50P-1
FIG. 50Q-1
FIG. 50R-1
FIG. 50S-1
FIG. 50T-1
FIG. 50U-1

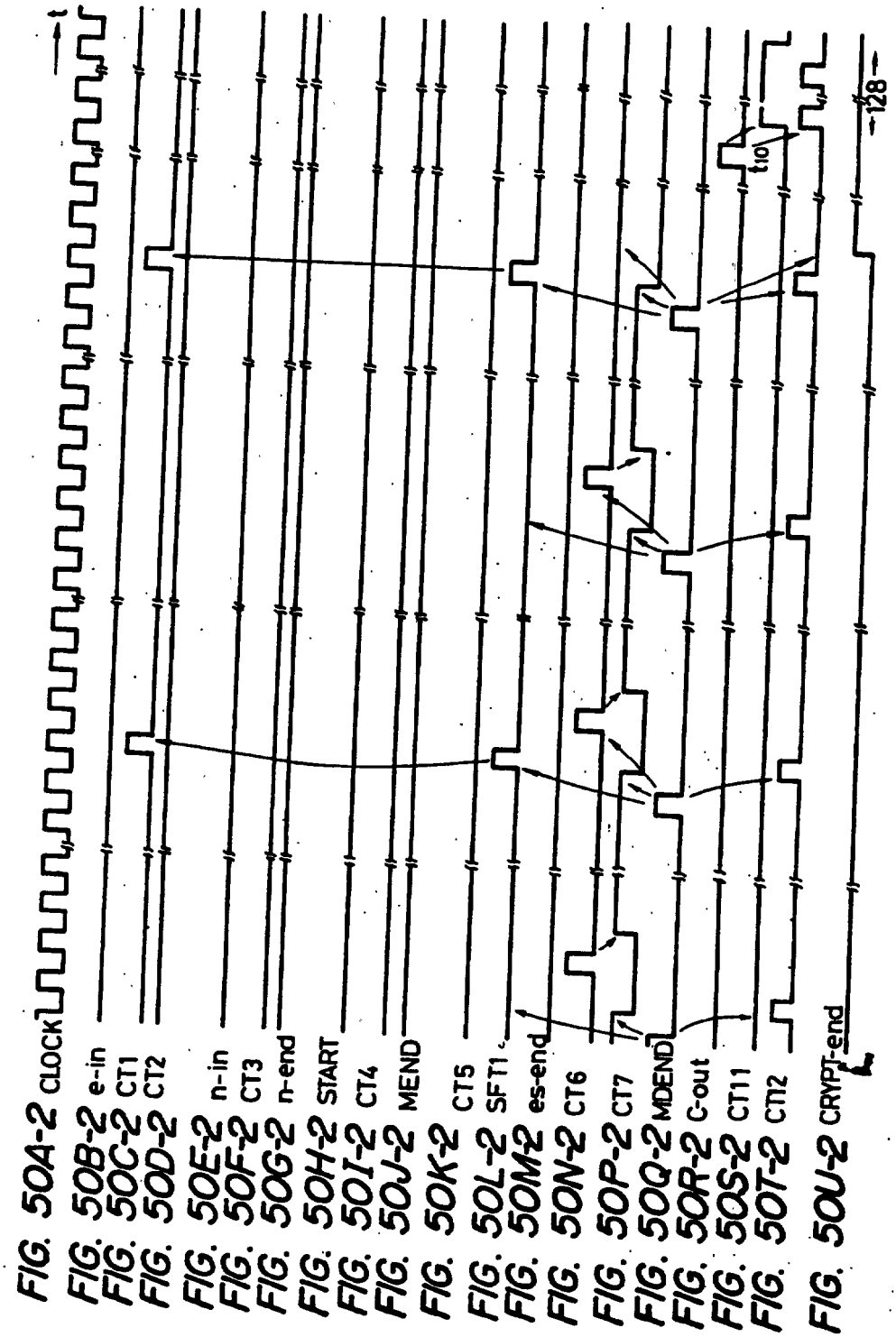


FIG. 50A-2 CLOCK
FIG. 50B-2 e-in
FIG. 50C-2 CT1
FIG. 50D-2 CT2
FIG. 50E-2 n-in
FIG. 50F-2 CT3
FIG. 50G-2 n-end
FIG. 50H-2 START
FIG. 50I-2 CT4
FIG. 50J-2 MEND
FIG. 50K-2 CT5
FIG. 50L-2 SFT1
FIG. 50M-2 es-end
FIG. 50N-2 CT6
FIG. 50P-2 CT7
FIG. 50Q-2 MDEND
FIG. 50R-2 C-out
FIG. 50S-2 CT11
FIG. 50T-2 CT12
FIG. 50U-2 CRYPT-end
t10
128

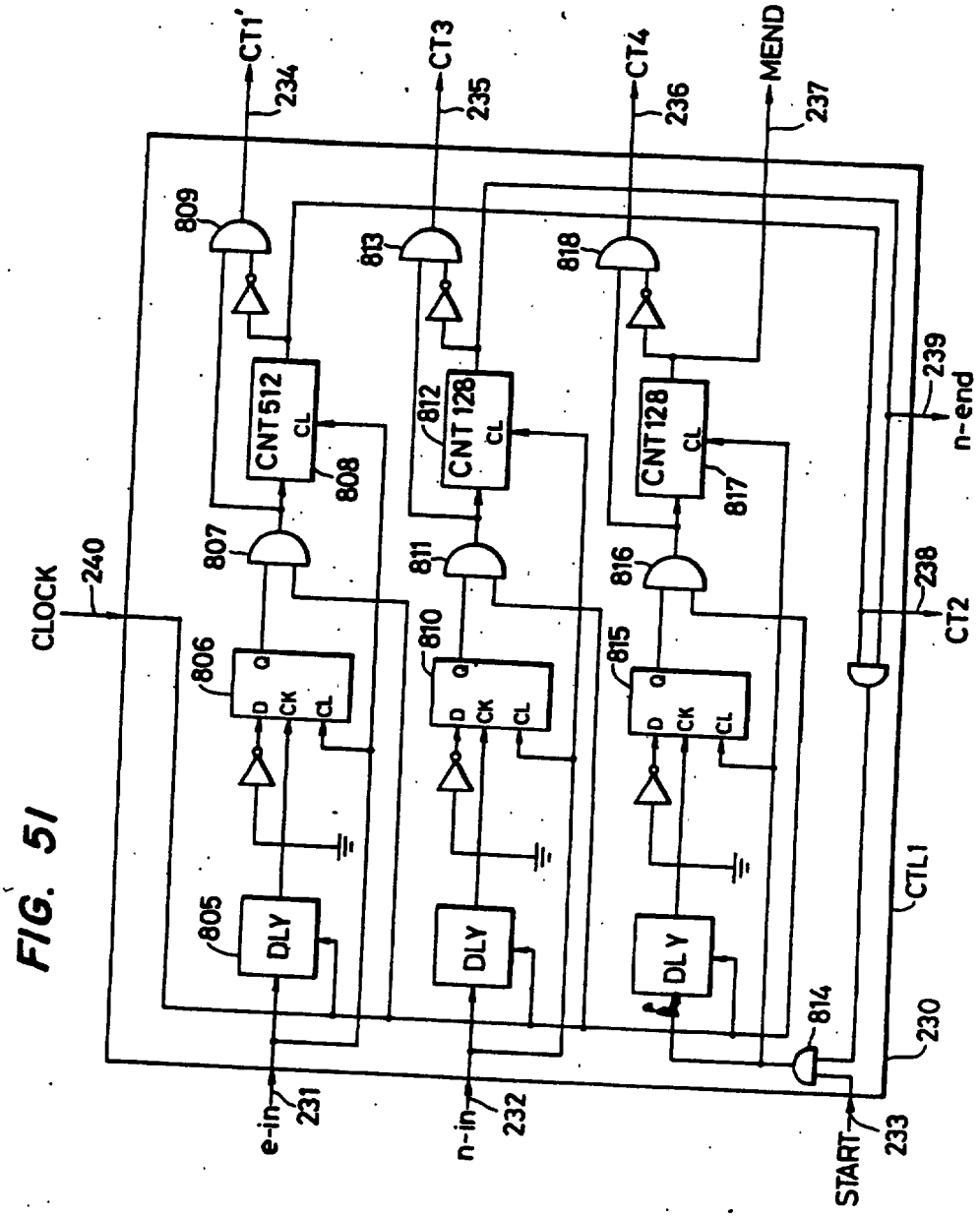

CT1'
234
CT3
235
CT4
236
MEND
237
809
813
818
CNT 512
CL
808
812
CNT 128
CL
CNT 128
CL
817
239
n-end
807
811
816
238
CT2
CLOCK
240
806
810
815
D
CK
CL
Q
DLY
805
CTL1
230
814
e-in
231
n-in
232
START
233

FIG. 51

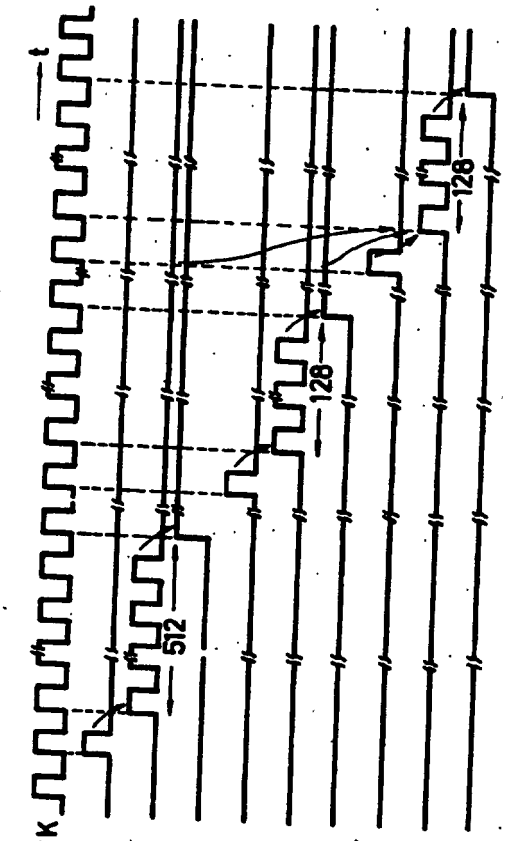


FIG. 52A
FIG. 52B
FIG. 52C
FIG. 52D
FIG. 52E
FIG. 52F
FIG. 52G
FIG. 52H
FIG. 52I
FIG. 52J

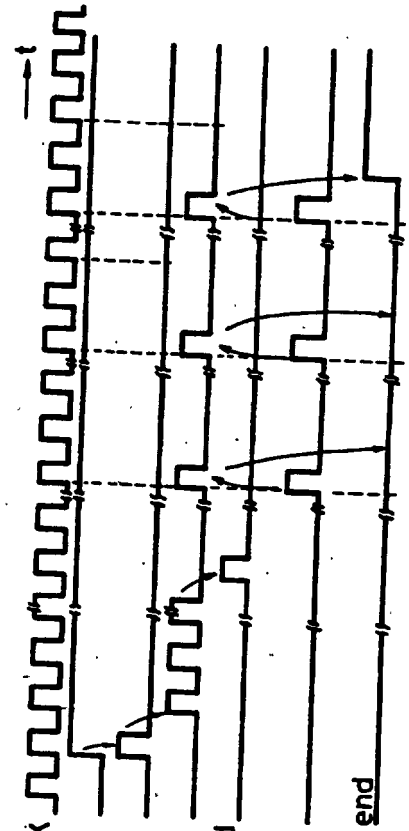


FIG. 54A
FIG. 54B
FIG. 54C
FIG. 54D
FIG. 54E
FIG. 54F
FIG. 54G

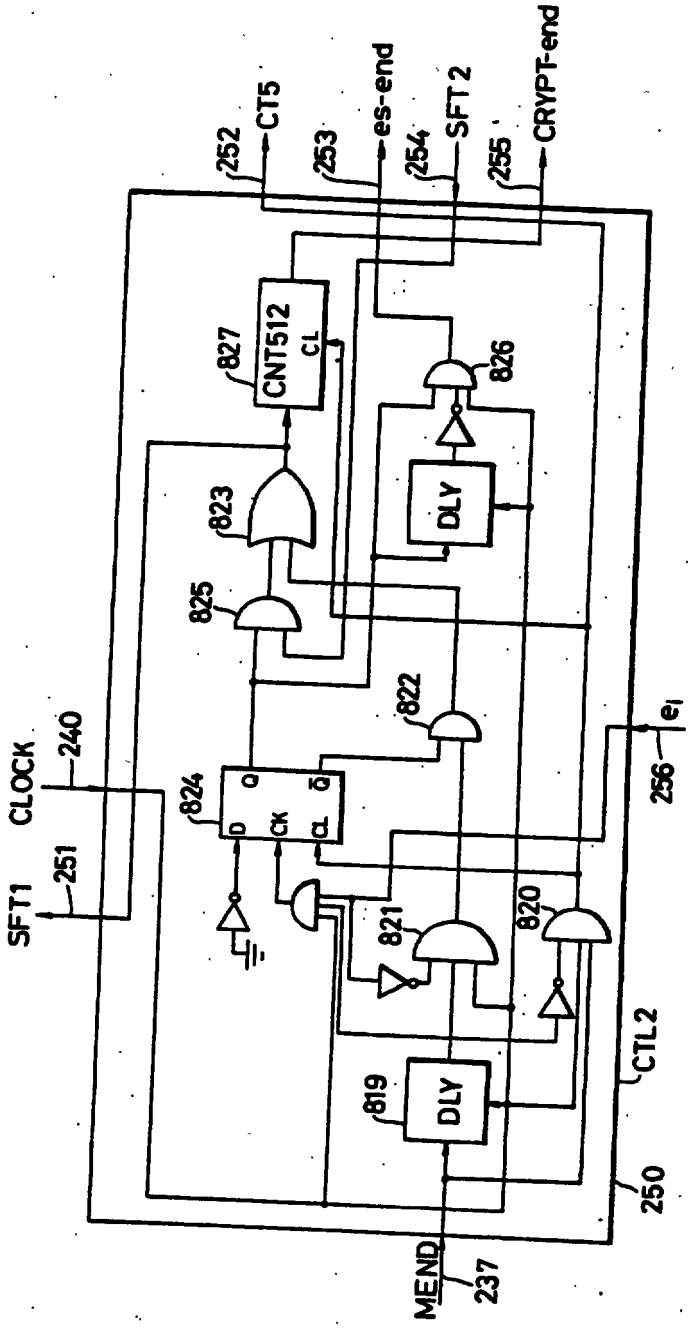
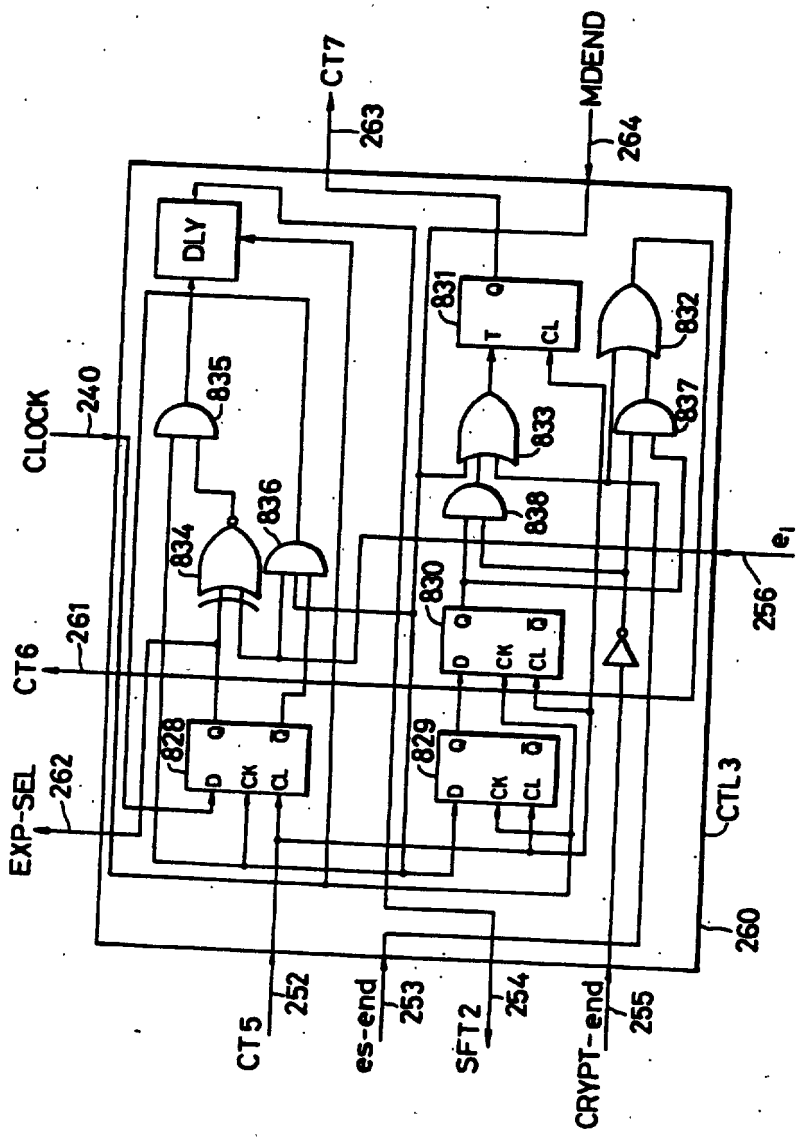

252
CT5
253
es-end
254
SFT 2
255
CRYPT-end
827
CNT512
CL
826
823
DLY
825
822
824
Q
D
CK
CL
240
CLOCK
251
SFT1
821
820
819
DLY
CTL2
250
MEND
237
256
e1

FIG. 53

FIG. 55


CT7
263
MDEND
264
DLY
831
832
CLOCK
240
835
833
837
836
834
838
830
e1
256
261
CT6
828
829
262
EXP-SEL
CTL3
260
252
253
254
255
CT5
es-end
SFT2
CRYPT-end

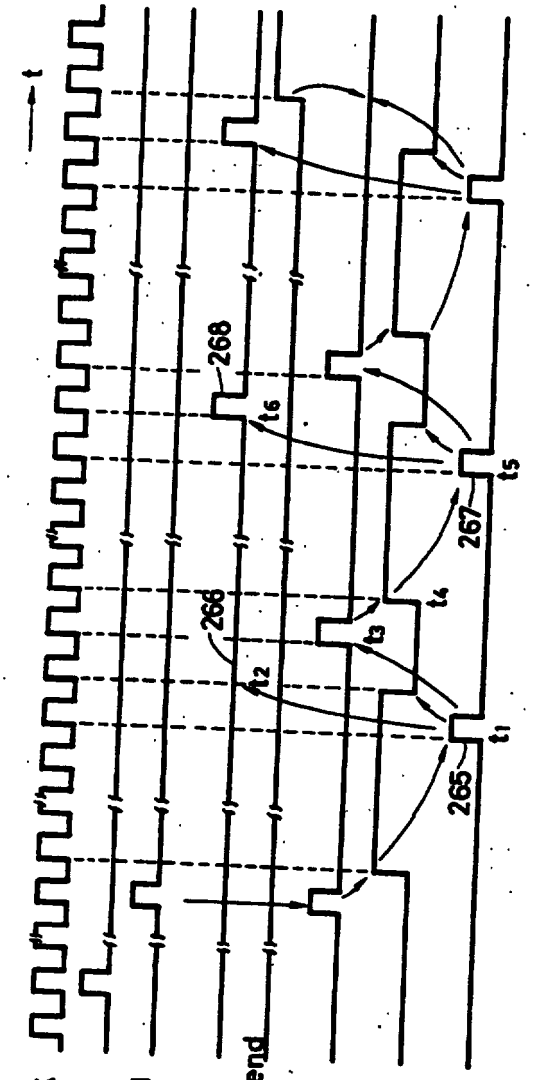
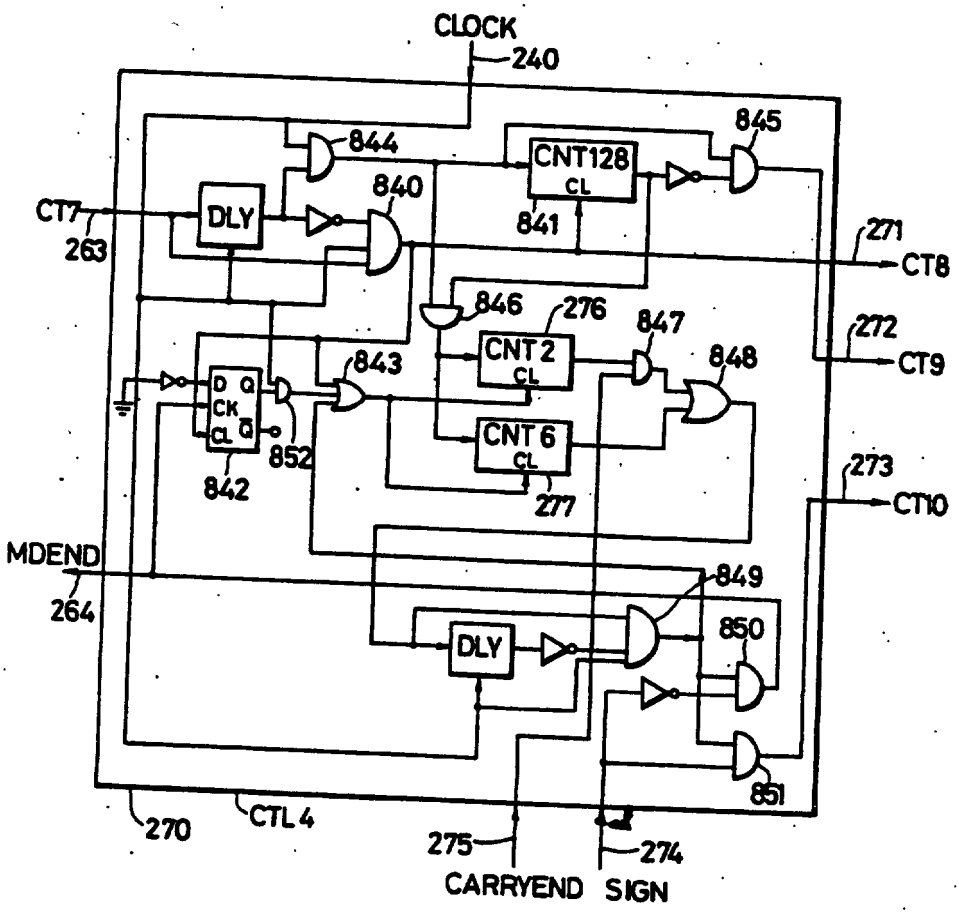


FIG. 56A
FIG. 56B
FIG. 56C
FIG. 56D
FIG. 56E
FIG. 56F
FIG. 56G
FIG. 56H

FIG. 57


CLOCK
240
844
845
CNT128
CL
840
DLY
CT7
263
841
271
CT8
846
276
847
272
CNT 2
CL
CT9
843
848
D Q
CK
CL Q
CNT 6
CL
852
842
277
273
CT10
MDEND
264
849
850
DLY
851
270
CTL 4
275
274
CARRYEND
SIGN

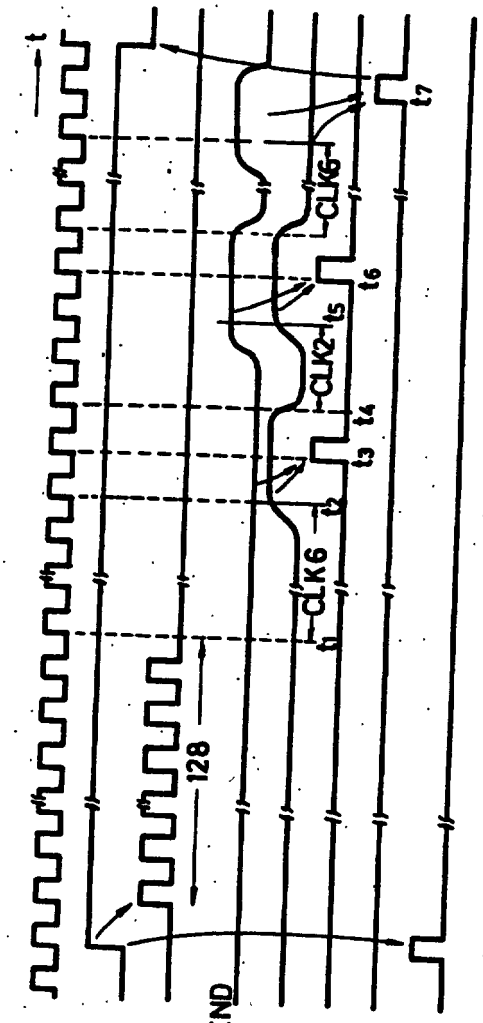


FIG. 58A
FIG. 58B
FIG. 58C
FIG. 58D
FIG. 58E
FIG. 58F
FIG. 58G
FIG. 58H

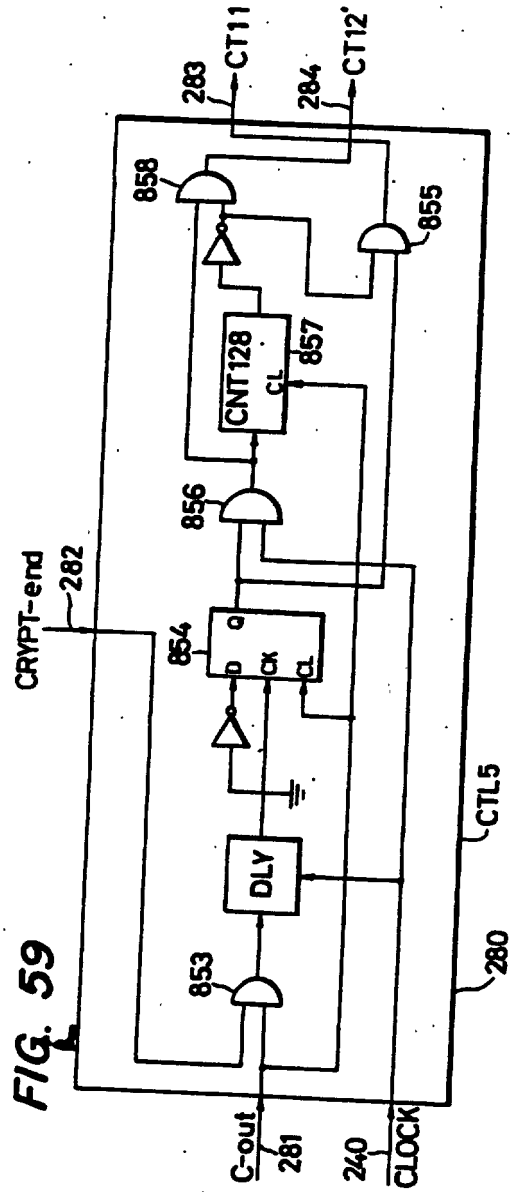


FIG. 59

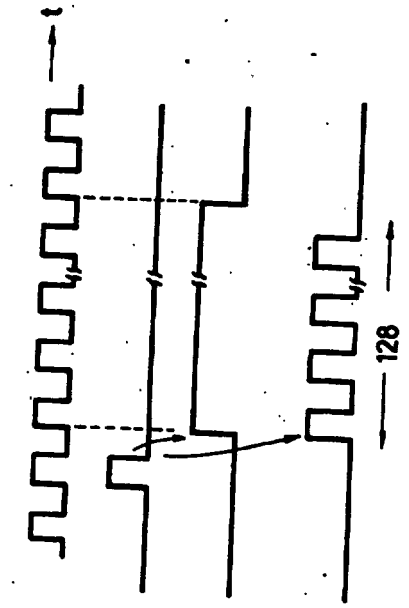


FIG. 60A CLOCK

FIG. 60B C-out

FIG. 60C CT11

FIG. 60D CT12'

FIG. 61

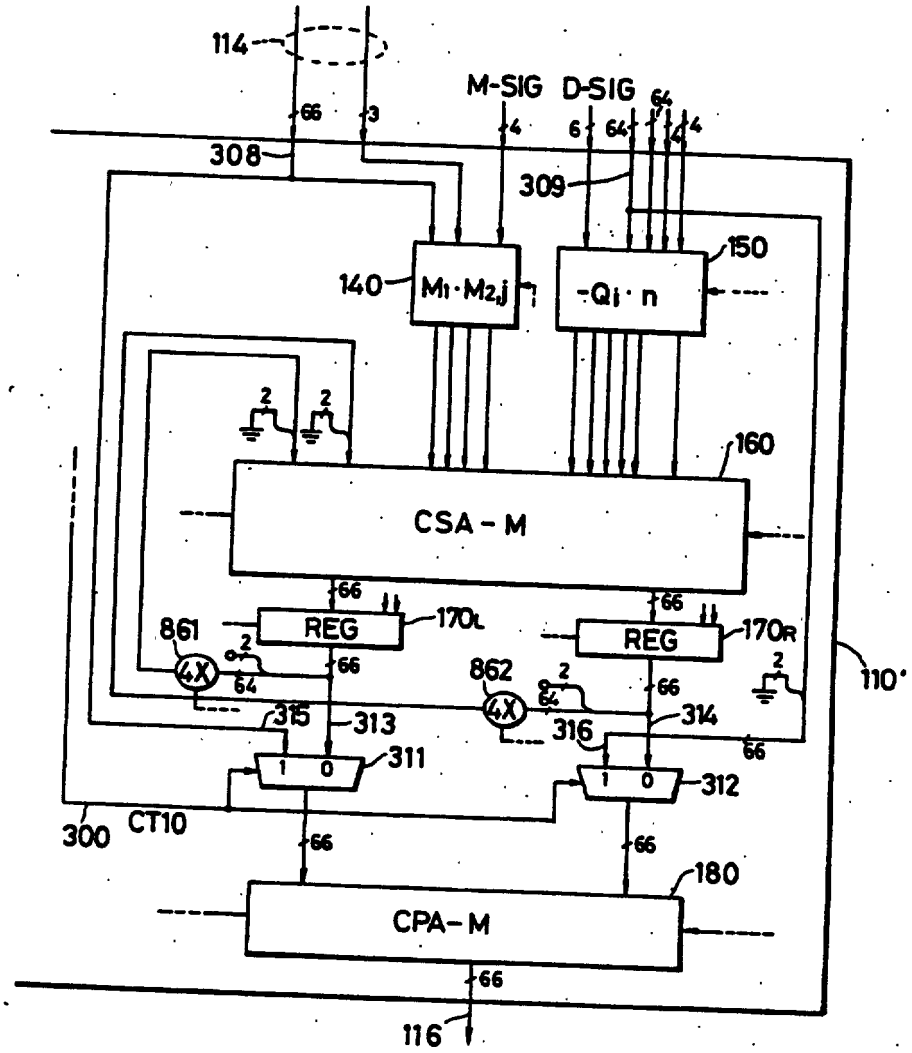

114
66
3
M-SIG
D-SIG
64
4
6
64
4
308
309
140
M1·M2j
-Qi·n
150
2
2
160
CSA-M
66
66
861
REG
170L
REG
170R
2
4X
64
66
862
2
4X
64
66
2
110'
315
313
316
314
66
1
0
311
1
0
312
300
CT10
66
66
180
CPA-M
66
116

FIG. 62

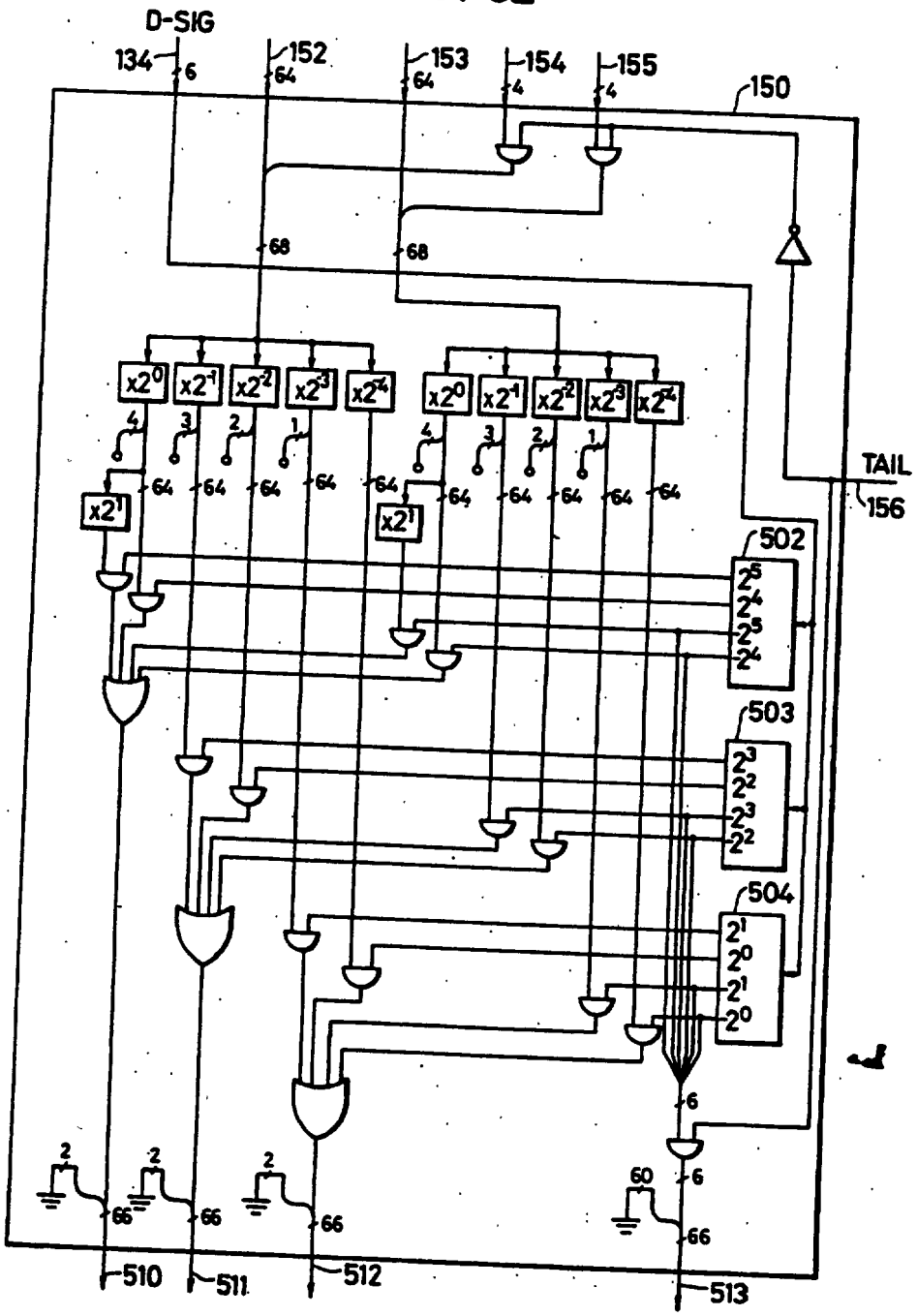

D-SIG
134
6
152
64
153
64
154
4
155
4
150
68
68
TAIL
156
502
503
504
510
511
512
513

FIG. 63

D-SIG INPUT			OUTPUT
qs	2^4	2^3	
0	0	0	0
0	0	1	2^4
0	1	0	2^4
0	1	1	2^5
1	0	0	0
1	0	1	-2^4
1	1	0	-2^4
1	1	1	-2^5

FIG. 64

D-SIG INPUT				OUTPUT
qs	2^3	2^2	2^1	
0	0	0	0	0
0	0	0	1	2^2
0	0	1	0	2^2
0	0	1	1	2^3
0	1	0	0	-2^3
0	1	0	1	-2^2
0	1	1	0	-2^2
0	1	1	1	0
1	0	0	0	0
1	0	0	1	-2^2
1	0	1	0	-2^2
1	0	1	1	-2^3
1	1	0	0	2^3
1	1	0	1	2^2
1	1	1	0	2^2
1	1	1	1	0

FIG. 65

D-SIG INPUT			OUTPUT
qs	2^1	2^0	
0	0	0	0
0	0	1	2^0
0	1	0	-2^1
0	1	1	-2^0
1	0	0	0
1	0	1	-2^0
1	1	0	2^1
1	1	1	2^0

FIG. 66

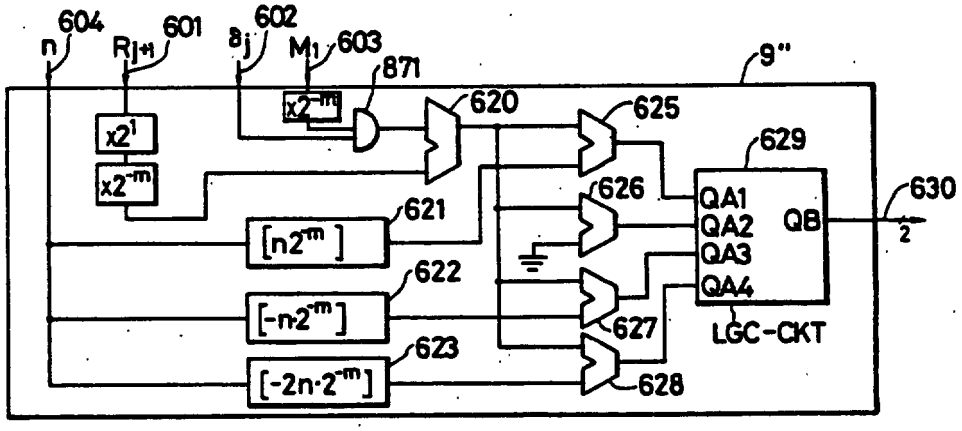


FIG. 67

QA1	0	0	0	0
QA2	1	0	0	0
QA3	1	1	0	0
QA4	1	1	1	0
QB	00	01	10	11

FIG. 68

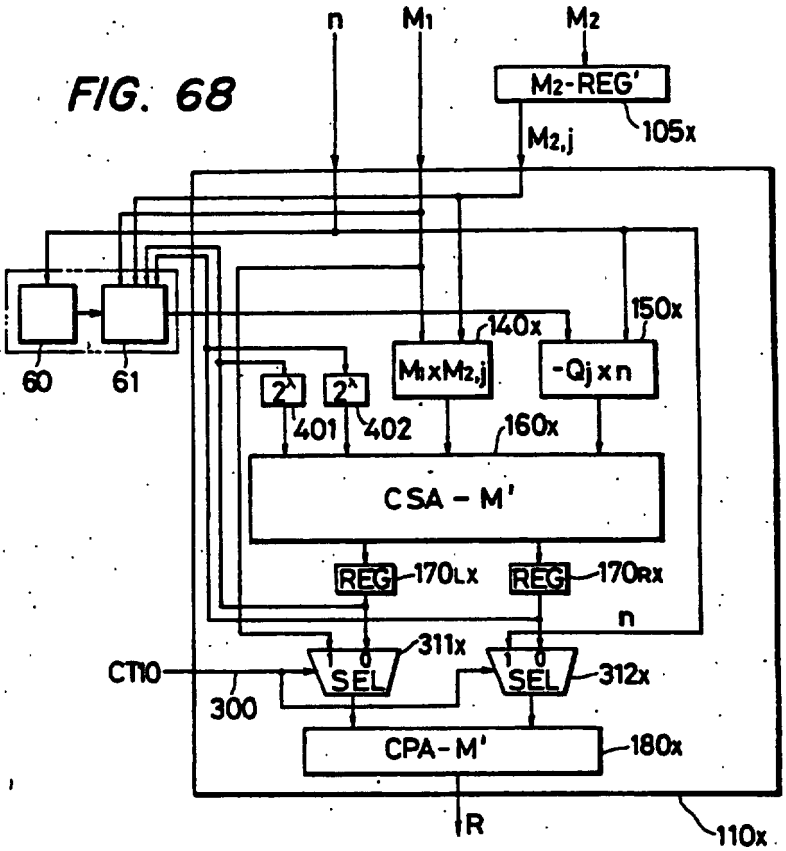

n
M1
M2
M2-REG'
105x
M2,j
150x
140x
60
61
M1xM2,j
-Qjxn
2^λ
2^λ
401
402
160x
CSA - M'
REG
170Lx
REG
170Rx
n
311x
CT10
SEL
SEL
312x
300
CPA-M'
180x
R
110x

FIG. 73

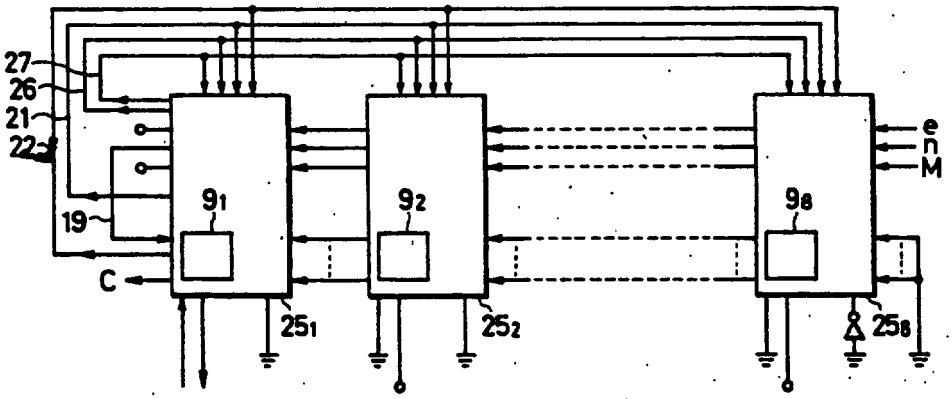

27
26
21
22
19
C
91
92
98
e
n
M
251
252
258

FIG. 69

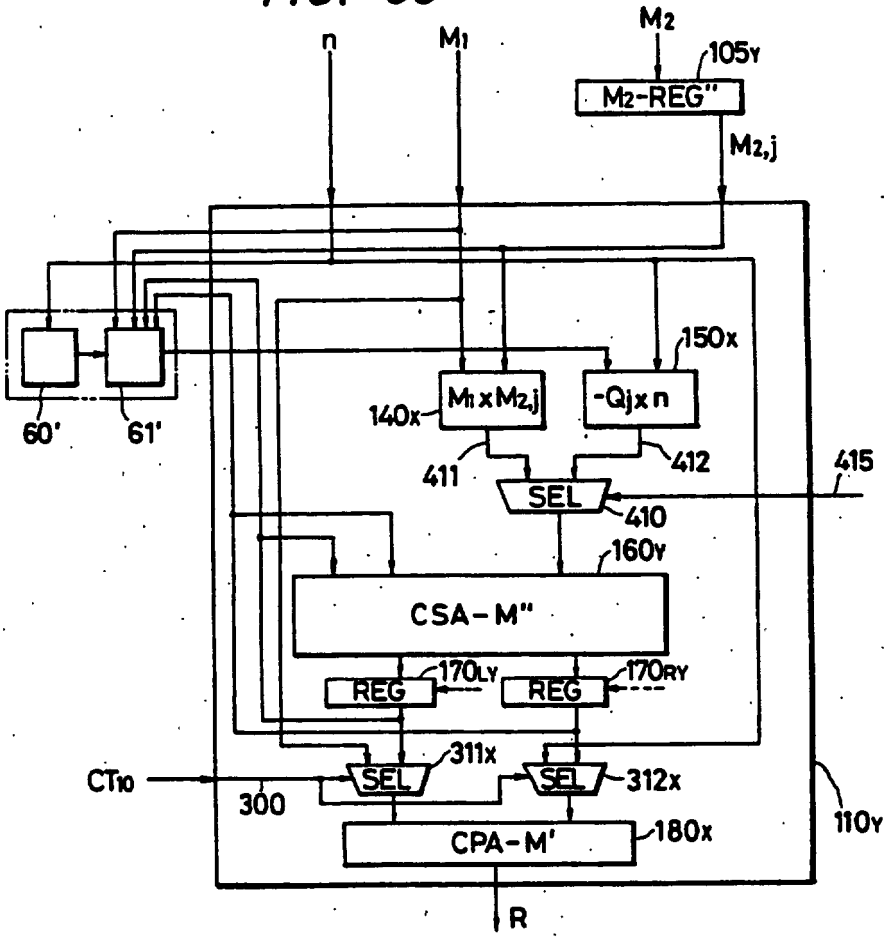

n
M1
M2
105Y
M2-REG"
M2,j
150x
60'
61'
140x
M1 x M2,j
-Qj x n
411
412
415
SEL
410
160Y
CSA-M"
REG
170LY
REG
170RY
CT10
300
SEL
311x
SEL
312x
CPA-M'
180x
110Y
R

FIG. 70

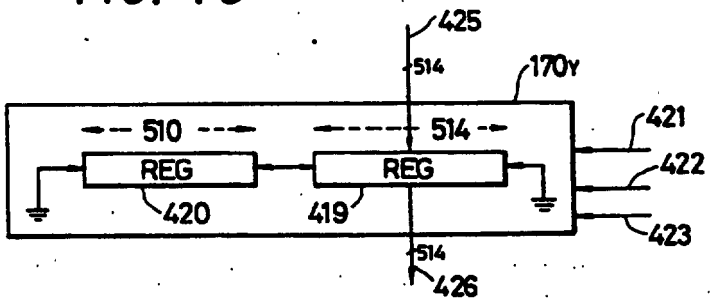

425
514
170Y
510
514
REG
REG
420
419
421
422
423
514
426

FIG. 71

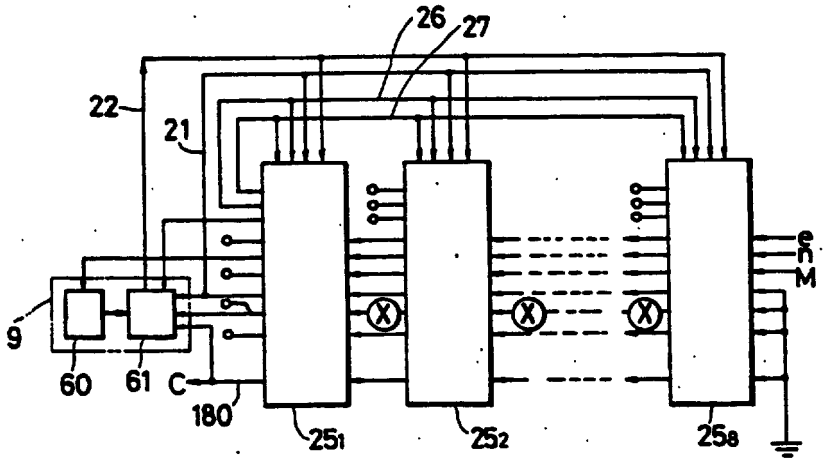
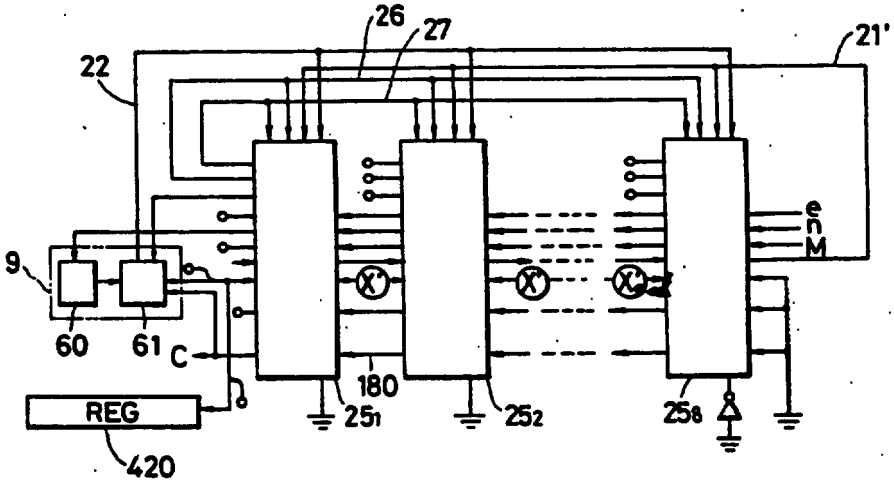

26
27
22
21
9
60
61
C
180
25₁
25₂
25₈
enM

FIG. 72


26
27
22
21'
9
60
61
C
REG
420
180
25₁
25₂
25₈
enM

CRYPTOSYSTEM

BACKGROUND OF THE INVENTION

The present invention relates to a cryptosystem for enciphering message or information used in ordinary communications and in electronic computers and deciphering the cryptogram and, more particularly, to a cryptosystem for encryption and/or decryption in a public-key cryptosystem in which an encryption key may be publicly revealed.

In the public-key cryptosystem, different keys are employed for encryption and decryption and anyone can encipher a message using a publicly revealed encryption key but only the receiver can decipher an enciphered message using a privately held decryption key, whereby to ensure privacy communications. Known as such a public-key cryptosystem is the RAS cryptosystem proposed in R. L. Rivest et al. "A Method for obtaining Digital Signatures and Public-Key Cryptosystems", Communications of the ACM, February 1978, Vol. 21, No. 2, pp 120–126.

An encryption and a decryption procedures are represented by the following congruence expressions:

$$\text{Encryption: } C \equiv M^e \bmod n \quad (1)$$

$$\text{Decryption: } M \equiv C^d \bmod n \quad (2)$$

where C, M, e, d and n are all integers, C a representation of a cryptogram as an integer, M a representation of a plain text as an integer, e and n an encryption key, d and n a decryption key and $e \neq d$. In the present invention all the variables except control signals are integers and are represented by 2's complement. The values of n, e and d are chosen, for enhancement of security protection capabilities, as follows: $n = 10^{100}$ to 10^{200}, $e = 10^{50}$ to 10^{100} and $d = 10^{50}$ to 10^{100}. The encryption procedure, i.e. a calculation of the remainder C when M^e is divided by n, is carried out in the manner described below. Here, M_1, M_2, R and C are variables. Preparation: Let e be represented by

$$e = \sum_{i=0}^{k} e_i \cdot 2^i$$

where $e_i = 0$ or 1.

Step 1: Set the variable C to 1.

Step 2: Execute steps 2a and 2b for i=k, k−1, . . . , 1, 0.

Step 2a:

$M_1 = C$, $M_2 = C$

$R \equiv M_1 \times M_2 \bmod n$

$C = R$

Step 2b:

When $e_i = 1$

$M_1 = C$, $M_2 = M$

$R \equiv M_1 \times M_2 \bmod n$

$C = R$

Step 3: Halt.

In the above steps the equation symbol "=" means to set the value of the right side to the variable of the left side.

Thus the encryption procedure of the RAS cipher, that is, computation of $C \equiv M^e \bmod n$, is completed. This calculating procedure will hereinafter be called an "exponentiation procedure".

As will be seen from comparison of Eqs. (1) and (2), the decryption procedure is similarly performed using d instead of e. In the case where the RAS cryptosystem which performs such encryption and decryption as described above should be implemented through utilization of the LSI technology as of CMOS, nMOS and so forth, the circuit scale of the cryptosystem would be on the order of 100 to 200 K gates. Since the integration density of prior art LSIs is in the range of 10 to 30 K gates per chip, implementation of such cryptosystem is difficult.

To avoid such difficulty, a crypto-LSI of a microprogram control system, having a circuit scale of about 20K gates, has been proposed in R. L. Rivest "A Description of a Single-Chip Implementation of the RSA Public-Key Cryptosystem", National Telecommunication Conference, 1980, Conference Record Vol. 3 of 4, pp 49.2.1–49.2.4. This crypto-LSI is impractical since its computing speed for cryptography is as low as 1.2K bits/s. Furthermore, since the encryption key of the RSA cryptosystem has a fixed length of 512 bits in this crypto-LSI, no procedure for cryptography can be carried out in the case where the length of the encryption key is, for example, 256- or 1024-bit.

As described above, in this cryptosystem, the calculation of $R \equiv M_1 \times M_2 \bmod n$ is conducted a number of times. In the past, this calculation has been performed in the same manner as ordinary multiplication and division; namely, $M_1 \times M_2$ is obtained by sequential multiplications in an ascending order starting with a least significant digit at first and then the multiplication result is divided by n sequentially in a descending order starting with a most significant digit. Therefore, this cryptosystem has the defect that the computing time is markedly long due to such sequential multiplication and division.

SUMMARY OF THE INVENTION

It is therefore an object of the present invention to provide a cryptosystem which can easily be fabricated as an LSI.

Another object of the present invention is to provide a cryptosystem which permits high-speed encryption and decryption.

Yet another object of the present invention is to provide a cryptosystem at low cost in which the length of an encryption and/or decryption key can be selected over a wide range, such as l bits (l being a constant), 2·l bits and a·l bits (a being an integer).

Since the encryption and the decryption are identical in procedure with each other as described previously, the following description will be given of the encryption procedure alone.

According to the present invention, the calculation in the aforementioned step

$$R \equiv M_1 \times M_2 \bmod n \quad (3)$$

is performed in the manner described below. The variables e, n, M, C, M_1 and M_2 are non-negative integers, and, in the following description, these characters are also used to represent signals respectively corresponding to the variables. For instance, the variable M_2 is a signal M_2, too, and a variable $\delta_{4(j-1)+i}$ (i=0, 1, 2, 3) is a signal $\delta_{4(j-1)+i}$ (i=0, 1, 2, 3), too. The variable M_2 is divided into l groups by steps of λ bits as follows:

$$M_2 = \sum_{j=1}^{l} M_{2j} \cdot 2^{(j-1)\lambda} \quad (4)$$

where j, R_j and Q_j are variables.

Step ① $R_{l+1}=0$

Step ② Set j=l, l−1, . . . 1 and perform the following operations:

$$Q_j=[(2^{\lambda}\cdot R_{j+1}+M_1\cdot M_{2j})\div n] \quad (5)$$

$$R_j=(2^{\lambda}\cdot R_{j+1}+M_1\cdot M_{2j})-Q_j\cdot n \quad (6)$$

Step 3 Halt. ($R_1=M_1\times M_2 \bmod n$)

Here, [x] represents the largest possible integer equal to or smaller than x. For instance, [1.0]=1, [1.5]=1, [−1.5]=−2 and so forth. By multiplying the both sides of Eq. (6) by $2^{(j-1)\lambda}$ and obtaining, for each side, the sum of the results of the multiplications for all j=1 to j=l as shown by the following equation, it is proved that this calculation method is correct.

$$\sum_{j=1}^{l} R_j \cdot 2^{(j-1)\lambda} = \sum_{j=1}^{l} 2^{(j-1)\lambda}\{2^{\lambda}\cdot R_{j+1} + M_1 \cdot M_{2j} - Q_j \cdot n\}$$

The addition and the subtraction in Eq. (6) can be performed at high speed using a carry save adder (CSA). Since the variables R_{j+1}, M_1 and n are extremely large, however, the calculation of Eq. (5) is liable to take too much time; therefore, it is preferred that the calculation of these equations be performed by using various approximations described hereinafter. Here, since the carry save adder has two outputs, R_j is divided into two as follows:

$$R_j = \sum_{i=0}^{1} R_{ji} \quad (7)$$

For high-speed calculation of Q_j, a constant of m bits is omitted from the low-order sides of all the variables in Eq. (5), and all the variables have been represented by a 2's complement as mentioned before. Q_j is approximated to Q_j' by the omission.

$$Q_j' = \left[\left(\sum_{i=0}^{1}[(2^{\lambda}\cdot R_{j+1,i})\cdot 2^{-m}] + \sum_{i=0}^{\lambda-1}[(M_1\cdot \delta_{(j-1)\lambda+i}\cdot 2^i)\cdot 2^{-m}] + S\right) + [n\cdot 2^{-m}]\right] \quad (8)$$

where

$$M_1 = \sum_{i=0}^{l\lambda-1} \delta_i \cdot 2^i,\ M_{2j} = \sum_{i=0}^{\lambda-1} \delta_{(j-1)\lambda+i}\cdot 2^i \quad (9)$$

Here, the constant S is introduced for suppressing any error resulting from the approximation.

Eq. (8) is a division, which takes much time. For speeding up the computation, a variable v for a reciprocal of the divisor $[n\cdot 2^{-m}]$ and a constant u are introduced, thereby to change Eq. (8) into a form of multiplication. By this procedure, Q_j' is approximated to Q_j''.

$$Q_j'' = [X_j'' \times v \times 2^{-u}] + t \quad (10)$$

-continued

where

$$\left.\begin{aligned} X_j'' &= \sum_{i=0}^{1}[(2^{\lambda}\cdot R_{j+1,i})\cdot 2^{-m}] + \sum_{i=0}^{\lambda-1}[(M_1\cdot \delta_{(j-1)\lambda+i}\cdot 2^i)\cdot 2^{-m}] + S \\ v &= [2^u \div [n\cdot 2^{-m}]] \\ t &= \begin{cases} 1 \text{ for } X_j'' \geq 0 \\ 0 \text{ for } X_j'' < 0 \end{cases} \end{aligned}\right\} \quad (11)$$

An error resulting from this close approximation cannot be made zero but can be reduced. By optimal selections of the constants m, S and u, errors γ_{1j} and γ_{2j} can be reduced as follows: The reason will be described later.

$$Q_j' = Q_j + \gamma_{1j},\ \gamma_{1j} = 0 \text{ or } 1 \quad (12)$$

$$Q_j'' = Q_j' + \gamma_{2j},\ \gamma_{2j} = 0 \text{ or } 1 \quad (13)$$

A concrete description will be given of the case of performing the operation $R=M_1\times M_2 \bmod n$ by the abovesaid close approximation. Since M and n are, for example, about 10^{200} which is roughly equal to 2^{500} as referred to previously, each variable is represented by a binary number of 512-bit length.

The following conditions are set, by way of example:

$$\left.\begin{aligned} &2^{511} < n < 2^{512} \\ &0 \leq M_1 < n \\ &0 \leq M_2 < n \\ &M_2 = \sum_{i=0}^{511} \delta_i \cdot 2^i,\ \delta_i = 0 \text{ or } 1 \\ &M_{2j} = \sum_{i=0}^{3} \delta_{4(j-1)+i}\cdot 2^i\ j = 128 \text{ to } 1 \\ &M_2 = \sum_{j=1}^{128} M_{2j}\cdot 2^{(j-1)\lambda} \\ &u = 13,\ m = 504,\ S = 38,\ \lambda = 4 \end{aligned}\right\} \quad (14)$$

(i) n is inputted and v is obtained from Eq. (11).

$$v \leftarrow [2^{13} \div [n\cdot 2^{-504}]] \quad (15)$$

where $2^5 < v < 2^6$.

(ii)

M_1 and M_2 are inputted. (16)

Repeated Calculation.

The calculation method will be shown below in the form of a program flowchart.

Step 0:

$$j \leftarrow 128,\ R_{129,1} \leftarrow 0,\ R_{129,0} \leftarrow 0 \quad (17)$$

Step 1: From Eq. (11)

$$X_j'' \leftarrow \sum_{i=0}^{1}[(R_{j+1,i})\cdot 2^{-500}] + \quad (18)$$

-continued

$$\sum_{i=0}^{3} [(M_1 \cdot \delta_{4(j-1)+i}) \cdot 2^i \cdot 2^{-504}] + 38$$

where $-2^{13} < X_j'' < 2^{13}$

Step 2:

$$\left.\begin{array}{l} Q_j'' \leftarrow [X_j'' \times v \times 2^{-13}] + 1 \text{ for } X_j'' \geq 0 \\ Q_j'' \leftarrow [X_j'' \times v \times 2^{-13}] \quad \text{for } X_j'' < 0 \end{array}\right\} \quad (19)$$

When $Q_j''=32$, set $Q_j''=31$ and when $Q_j''=-32$, set $Q_j''=-31$.

Step 3: From Eq. (6)

$$\sum_{i=0}^{1} R_{j,i} \leftarrow \sum_{i=0}^{1} 2^i \cdot R_{j+1,i} + M_1 \cdot M_{2,j} - Q_j'' \cdot n \quad (20)$$

Step 4:

$$\left.\begin{array}{l} \text{If } j = 1, \text{ then go to Step 5,} \\ \text{if } j \neq 1, \text{ then } j \leftarrow j - 1, \text{ and go back to Step 1.} \end{array}\right\} \quad (21)$$

Step 5: The repeated calculation ends.

CALCULATION FOR COMPENSATION

$$\left.\begin{array}{l} \text{Step 6: } R_1 \leftarrow \sum_{i=0}^{1} R_{1,i} \\ \text{If } R_1 \geq 0, \text{ then go to Step 8} \end{array}\right\} \quad (22)$$

$$\left.\begin{array}{l} \text{Step 7: } \sum_{i=0}^{1} R_{1,i} \leftarrow \sum_{i=0}^{1} R_{1,i} + n \\ \text{Go back to Step 6} \end{array}\right\} \quad (23)$$

Step 8: $R \leftarrow R_1$, end. (24)

In the case where the variable e is represented by 512 bits, 0 goes in succession on its high-order bit side. This arises from the aforesaid conditions $n=10^{100}$ to 10^{200}, $e=10^{50}$ to 10^{100}. Since j=128 to 1, it is seen that the repeated calculation is conducted 128 times. The range of Q_j'' obtained from the equation (19) is given by $-31 \leq Q_j'' \leq 31$. The calculation method mentioned above will be proved to be appropriate, later.

In the compensating calculation, the number of executions of Step 7 may be zero, one or two. The reason for this will be described later. At the time when Step 6 is executed for the first time, the following condition holds:

$$-2n \leq \sum_{i=0}^{1} R_{1,i} < n$$

So, a register of 514-bit length, including sign bit, is employed for storing $R_{1,i}$. Accordingly, an adder of a 514 bit width is used for performing the operation of Eq. (20). In the operation of Eq. (18), 500 bits are discarded for $R_{j+1,i}$ and 504 bits, 503 bits, 502 bits and 501 bits are discarded for M_i in accordance with the values i=0,1,2,3, respectively. An adder for obtaining X_j'' may be an adder of 14-bit width, including sign bit, because of the condition $2^{-13} < X_j'' < 2^{13}$.

As described previously, the operation $C=M_1 \times M_2 \bmod n$ necessary for the calculation for cryptography can be performed by eight steps ① to ⑧. Embodiments of the present invention, described later, execute such a computation. That is, a quotient calculating unit, a main adding unit and a controller are provided. To the quotient calculating unit are applied M_1, $M_{2,j}$, n and R_{j+1} to perform an operation $Q_j=[(M_1 \times M_{2,j}+2^\lambda \cdot R_{j+1}) \div n]$. To the main adding unit are provided M_1, $M_{2,j}$, Q_j, R_{j+1} and n to conduct an operation $M_1 \times M_{2,j}+2^\lambda R_{j+1}-Q_j n$. The controller controls the quotient calculating unit and the main adding unit so that these operations are performed in the order j=l, l−1, . . . 1. That is, as indicated by the order j=l, l−1, . . . 1, the operation $M_1 \times M_2 \bmod n$ is performed by simultaneously carrying out multiplication and division in a descending order, so that the calculation is conducted at high speed. Furthermore, the calculation in the quotient calculating unit can be further speeded up by discarding and multiplication based on Eq. (10). By using the carry save adder, the addition and subtraction in the main adding unit can be speeded up by the time necessary for carry propagation. This is very significant because the numbers of digits of M and n are very large and because the number of calculations is large.

In the present invention, the main adding unit is divided into a plurality of slice sections of the same function. To the slice sections are sequentially applied M_1 and n while being divided for each constant width of their binary integers, and $M_{2,j}$ and Q_j are provided to the slice sections in common to them. For each set of M_1, n, Q_j, $M_{2,j}$ and R_{j+1}, an operation $R_j=M_1 \times M_{2,j}+2^\lambda \cdot R_{j+1}-Q_j n$ is performed. The slice sections are connected in cascade via signal lines so that a part of each calculation result may be provided to a higher order slice section. In each slice section, one or more registers for storing divided portions of M, n, e, R_j and C are provided as required. By such division of the main adding unit into slice sections, each slice section can easily be fabricated as an LSI even by the present LSI technology, so that the cryptosystem can be produced at low cost. Moreover, by increasing or decreasing the number of slice sections, the lengths of the encryption and decryption keys e and d can be varied with ease.

By applying such division of the main adding unit into slice sections to the case where $M_1 \cdot M_2 \bmod n$ is calculated by performing the multiplication $M_1 \cdot M_2$ prior to the division by n, the cryptosystem can be fabricated at low cost.

BRIEF DESCRIPTION OF THE DRAWINGS

FIG. 1 is a block diagram showing the principle of a conventional technology for the RSA cryptosystem;

FIG. 2 is a block diagram showing the principle of the cryptosystem of the present invention;

FIG. 3 is a block diagram showing the principle of dividing a main adding unit;

FIGS. 4A to 4Z and 5A to 5Q illustrate symbols of various functions used in the drawings;

FIG. 6 is a block diagram illustrating the whole arrangement of an embodiment of the present invention;

FIG. 7 is a block diagram showing an example of a quotient calculating pre-processing section 60 used in FIG. 6;

FIG. 8 is a block diagram showing an example of a quotient calculating post-processing section 61 used in FIG. 6;

FIG. 9 is a diagram illustrating a specific example of an AND element group 70 used in FIG. 8;

FIG. 10 is a diagram illustrating a specific example of a constant generator 71 utilized in FIG. 8;

FIG. 11 is a diagram illustrating a specific example of an adder 72 employed in FIG. 8;

FIG. 12 is a diagram showing a specific example of a carry save adder CSAUQ1 used in FIG. 8;

FIG. 13 is a diagram showing an example of an adder 73_2 used in FIG. 8;

FIG. 14 is a diagram showing an example of an adder 74_1, used in FIG. 8;

FIG. 15 is a block diagram illustrating a specific example of a slice section employed in FIG. 6;

FIG. 16 is a diagram illustrating an example of an M register 101 utilized in FIG. 15;

FIG. 17 is a diagram showing an example of an n register 103 employed in FIG. 15;

FIG. 18 is a diagram illustrating an example of a C register 104 used in FIG. 15;

FIG. 19 is a diagram showing an example of an M_2 register 105 employed in FIG. 15;

FIG. 20 is a diagram illustrating an example of an e register 102 used in FIG. 15;

FIG. 21 is a diagram illustrating an example of a selector 106 utilized in FIG. 15;

FIG. 22 is a diagram showing an example of a main adding unit 110 employed in FIG. 15;

FIG. 23 is a diagram showing a specific example of an $M_1 \cdot M_2$ calculating section 140 used in FIG. 22;

FIG. 24 is a diagram showing a specific example of a $-Q_i n$ calculating section 150 used in FIG. 22;

FIG. 25 is a diagram illustrating a specific example of an adding section 160 used in FIG. 22;

FIG. 26 is a diagram illustrating a specific example of a carry save adder 161 used in FIG. 25;

FIG. 27 is a diagram illustrating an example of a register section 170_L utilized in FIG. 22;

FIG. 28 is a diagram showing an example of an adder 180 employed in FIG. 22;

FIG. 29 is a diagram showing an example of a carry detector 190 used in FIG. 22;

FIG. 30 is a diagram showing the coupling state of the M register 101;

FIG. 31 is a diagram showing the coupling state of the e register 102;

FIG. 32 is a diagram showing the coupling state of the n register 103;

FIG. 33 is a diagram showing the coupling state of the C register 104;

FIG. 34 is a diagram showing the coupling state of the M_2 register;

FIG. 35 is a diagram showing the coupling state of the selector 106;

FIG. 36 is a diagram showing the coupling state of the main adding unit 110;

FIG. 37 is a diagram showing the coupling state of the $M_1 \cdot M_2$ calculating section 140;

FIG. 38 is a diagram showing the coupling state of the $-Q_i n$ calculating section;

FIG. 39 is a diagram showing the coupling state of the adding section 160;

FIG. 40 is a diagram showing the coupling section of the register section 170_L;

FIG. 41 is a diagram showing the coupling state of the adder 180;

FIG. 42 is a diagram showing the coupling state of the carry detector 190;

FIG. 43 is explanatory of an operation in the coupling states depicted in FIGS. 39 to 41;

FIG. 44 is a diagram showing an arrangement of bits in the coupling state depicted in FIG. 37;

FIG. 45 is explanatory of the operation in the coupling state depicted in FIG. 37;

FIG. 46 is explanatory of an operation in the coupling state depicted in FIG. 38;

FIG. 47 is explanatory of an operation in the coupling state depicted in FIG. 40;

FIG. 48 is explanatory of an operation in the coupling state depicted in FIG. 42;

FIG. 49 is a block diagram showing the outline of a controller 8;

FIGS. $50A_1$ to $50U_1$ and FIGS. $50A_2$ to $50U_2$ are, as a whole, a timing chart illustrating the outline of the operation of the controller 8 used in FIG. 6;

FIG. 51 is a diagram illustrating a specific example of a first control section 230 in the controller 8;

FIGS. 52A to 52J are, as a whole, a timing chart showing the operation of the first control section 230;

FIG. 53 is a diagram illustrating a specific example of a second control section 250 in the controller 8;

FIGS. 54A to 54G are, as a whole, a timing chart showing the operation of the second control section 250;

FIG. 55 is a diagram illustrating a specific example of a third control section 260 in the controller 8;

FIGS. 56A to 56H are, as a whole, a timing chart showing the operation of the third control section 260;

FIG. 57 is a diagram illustrating a specific example of a fourth control section 270 in the controller 8;

FIGS. 58A to 58H are, as a whole, a timing chart showing the operation of the fourth control section 270;

FIG. 59 is a diagram illustrating a specific example of a fifth control section 280 in the controller 8;

FIGS. 60A to 60D are, as a whole, a timing chart showing the operation of the fifth control section 280;

FIG. 61 is a diagram illustrating a modified form of the embodiment of FIG. 6 in which the main adding unit 110 is coupled and used as another means for compensating calculation;

FIG. 62 is a diagram illustrating another example of the $-Q_i n$ calculating section in FIG. 22;

FIGS. 63 to 65 are diagrams respectively showing the logic of circuits 502 to 504 in FIG. 62;

FIG. 66 is a diagram illustrating another example of a quotient calculating unit 9;

FIG. 67 is a diagram showing the logic of a circuit 629 in FIG. 66;

FIG. 68 is a block diagram illustrating the main adding unit in the cryptosystem in the case where the multiplication and division are performed at the same time;

FIG. 69 is a block diagram illustrating the main adding unit in the cryptosystem in the case where the multiplication and division are performed one after the other;

FIG. 70 is a diagram illustrating a specific example of a register section 170_Y in FIG. 69;

FIG. 71 is a diagram illustrating an example of the main adding unit shown in FIG. 68 being divided;

FIG. 72 is a diagram illustrating an example of the main adding unit shown in FIG. 69 being divided; and

FIG. 73 is a diagram illustrating another embodiment of the present invention where the quotient calculator 9 is provided in each of the slice sections.

DESCRIPTION OF THE PREFERRED EMBODIMENTS

To facilitate a better understanding of the present invention, a description will be given first of a conventional technology for the RSA cryptography. FIG. 1 shows the principle of a conventional technology which performs calculations for the RSA cryptography. An M-register 1, an e-register 2, an n-register 3 and a C-register 4 are provided for storing variables M, e, n and C, respectively. The contents of the M-register 1 and the C-register 4 are supplied to a selector 6 via signal lines 12 and 11, respectively. The selector 6 selects one of the signals from the signal lines 11 and 12 in accordance with a switching signal from a switching signal line 13 and provides the selected signal to an M_2-register 5. A multiplier-divider 7 is supplied with a signal M_1 on a signal line 14, a signal M_2 on a signal line 15 from the M_2-register 5 and a signal n on a signal line 17 from the n-register 3. The most significant bit (MSB) of the e-register 2 is provided via a signal line 18 to a controller 8, which, in turn, controls the selector 6 in accordance with the content of the signal e applied. The signal lines are each composed of a plurality of signal conductor lines.

At first, the variables M, e and n are stored in the registers 1, 2 and 3, respectively. The e-register 2 is such one that has a left circular shift function. Prior to the exponentiation procedure, the content of the e-register 2 is shifted to left until the left-most bit of the e-register becomes "1". The reason is that the number of calculations in steps 2a and 2b of the exponentiation procedure can be reduced by starting the calculation with a condition $e_i=1$.

Then the controller 8 stores +1 into the C-register 4. Represented by C, the content of the C-register 4 is C=1. The above is the operation of step 1 of the exponentiation procedure.

Next, the controller 8 executes steps 2a and 2b of the exponentiation procedure in the following manner:

On the input signal line 17 of the multiplier-divider is always provided the variable n. Let the signals on the input signal lines 14 and 15 of the multiplier-divider 7 be represented by M_1 and M_2, respectively, and a signal on an output signal line 16 of the multiplier-divider 7 be represented by R. Since the C-register 4 is connected to the signal line 14, $M_1 \leftarrow C$ is executed. The selector 6 selects the input signal line 11 in accordance with the signal on the signal line 13 from the control unit 8, and the content C of the C-register 4 is latched in the M_2-register 5. Accordingly, the signal M_2 on the signal line 15 becomes $M_2 \leftarrow C$. Then, the multiplier-divider 7 performs the operation $R=M_1 \times M_2 \bmod n$ and provides the signal R on the output signal line 16, so that the content of the C-register 4 becomes R, thus executing $C \leftarrow R$. The above is the operation of step 2a of the exponentiation procedure.

The operation of step 2b of the exponentiation procedure differs from the operation of step 2a only in the operation of the selector 6. That is, the input signal line 12 of the selector 6 is selected and the content of the M register 1 is latched in the M_2-register 5, resulting in $M_2 \leftarrow M$.

The controller 8 executes the operations of steps 2a and 2b while shifting the content of the register 2 to left bit by bit for each e_i of the variable

$$e = \sum_{i=0}^{k} e_i \cdot 2^i.$$

By such operation, the content C of the C-register 4 finally becomes $C=M^e \bmod n$ based on the exponentiation procedure. By the way, the principle of the calculation order of the RSA cryptosystem shown in FIG. 1 is known, but the construction of the multiplier-divider used therein has not been disclosed and the cryptosystem has not been put to practical products.

FIG. 2 is explanatory of the principle of the cryptosystem of the present invention, the parts corresponding to those in FIG. 1 being identified by the same reference numerals. In FIG. 2, the multiplier-divider used in FIG. 1 is divided into a quotient calculator 9 and a main adder 10. The quotient calculator 9 performs the operation of Eq. (5), i.e. the division for obtaining the quotient, using Eqs. (18) and (19). The main adder 10 is formed of the remaining portion of the multiplier-divider 7 from which is separated the quotient calculator 9, and it mainly performs the additions in Eqs. (20) and (23). That is, in the main adder 10, for example, as shown in Eq. (20), the multiplication and division are simultaneously performed in a descending order starting with the most significant digit, permitting high-speed computation. In the present invention, the quotient calculator 9 is separated from the multiplier-divider 7; this is one of features of the present invention which distinguishes it over the prior art. Since the quotient calculator 9 is separated, signal lines 19 to 24 for connecting the quotient calculator 9 to other parts are added. The part except for the quotient calculator 9 in FIG. 2, that is, the part identified by 25' will hereinafter be called a "sliceable section". It will be evident that, with such an arrangement as shown in FIG. 2, the calculation for cryptography can be conducted by the exponentiation procedure as is the case with the prior art example of FIG. 1. In this case, since Eqs. (5) and (6) are used, the signal M_2 becomes a signal M_{2j}.

FIG. 3 shows the principle of divisional arrangement of the cryptosystem of the present invention. In the sliceable section 25' in FIG. 2, the parts except for the controller 8 are each divided into, for example, eight and the eight groups are each provided with one controller 8 to constitute eight sliced sections 25_1 to 25_8. Here, the division into eight is to divide, for example, the M-register 1 of 512-bit length by 64 bits to form eight 64-bit registers 1_1 to 1_8, by which 512-bit information is represented. The registers 2, 3, 4 and 5 are respectively divided into registers 2_1 to 2_8, 3_1 to 3_8, 4_1 to 4_8 and 5_1 to 5_8. The selector 6 is similarly divided into eight. Also the main adder 10 in FIG. 2 is divided into eight and each processes 64-bit information divided from 512-bit information. Signal lines 26 and 27 are necessitated as a result of the division of the sliceable section 25'. Signal lines 28_1, 28_2 and 28_3 are input signal lines for the variables e, n and M, and a signal line 29 is an output signal line for the variable C. In this way, the sliceable section can easily be divided because the quotient calculator 9 is not included therein. The cryptosystem of an embodiment of the present invention based on the principle shown in FIG. 3, comprises a plurality of sliced sections obtained by equally dividing the sliceable sec-

tion 25' of the cryptosystem of FIG. 2 and combining each sliced section with the controller 8, and one quotient calculator. This arrangement has the following features.

The sliceable section 25' in FIG. 2 is difficult to fabricate as one chip through using the present LSI technology because it requires 100 to 200K gates when materialized as a hardware. According to this embodiment, however, the sliced sections 25_1 to 25_8 are each on the order of 15 to 30K gates, and hence can be implemented by the existing LSI technology. At the same time, since these sliced sections may be formed by the same type of chips, the number of processes involved in the design of the cryptosystem is small, reducing the manufacturing costs.

Furthermore, by increasing or decreasing the number of sliced sections, it is possible to implement cryptosystems at low cost which have various lengths of the encryption and/or decryption keys n and e. A description will be given later in this connection.

In the foregoing, the sliced sections 25_1 to 25_8 are each described as to handle 64-bit information but, strictly speaking, the main adder handles 66 bits and 64 bits of them are used for the purpose described previously. This will be described later. The quotient calculator 9 can be divided into a quotient pre-processing section and a quotient post-processing section in accordance with the nature of its processing. The signal lines 26, 21 and 22 will hereinafter be referred to as the exponentiation control signal line, the multiplication control signal line and the division control signal line, respectively.

SYMBOLIZATION CONVENTIONS

Prior to a detailed description of the invention, a description will be given of symbols used for showing various functions in the drawings.

FIG. 4A shows that a terminal 30 of a signal line is not connected to any parts, that is, an open terminal. Incidentally, the signal line is usually composed of more than one signal lines and, in this case, the open terminal represents plural open terminals. FIG. 4B shows that a+b signal lines (a=1, 2, . . . and b=1, 2, . . .) are branched into a and b signal lines. In this case, the left-hand a signal lines transmit an a-digit signal from the most significant digit side of the a+b signal lines, represented as a binary number, whereas the right-hand b signal lines similarly transmit a b-digit signal from the least significant side. The arrows of the signal lines indicate the direction of signal transmission. This is common to all the accompanying drawings. When the branching is indicated by lateral lines as seen in FIG. 4C, the upper side indicates the higher-order digit. That is, in the case of a signal represented as a binary number, the signal line on the right-hand side in the direction of the signal transmission indicates the most significant digit and the signal line on the left side the least significant digit. FIG. 4D shows that b groups of signal lines, each including a lines, are described en bloc.

FIG. 4E shows an AND logic of two inputs. This also applies to an AND logic of three or more inputs.

FIG. 4F shows a NAND logic of two inputs. This also applies to a NAND logic of three or more inputs. FIG. 4H shows an exclusive OR and FIG. 4I NOT of the exclusive OR. FIG. 4J shows NOT logic. FIG. 4K shows that a signal value "0" is provided. FIG. 4L shows that a signal value "1" is provided. FIG. 4M shows a one-bit full adder. Letting signals on signal lines 31_1, 31_2 and 31_3 be represented by A, B and C (A=0,1, B=0,1, C=0,1), respectively, data on the signal lines 32_1 is indicated by $A \oplus B \oplus C$ (where $\oplus$ is exclusive OR) and data on the signal line 32_2 is $A \cdot B + B \cdot C + C \cdot A$ (where · is AND and + is OR).

FIG. 4N shows a two-input selector. Two input signal lines 34_0 and 34_1 and an output signal line 35 are all composed of a (a=1, 2, . . .) signal lines. When a selector input switching signal on a selector input signal switching signal line 33 is 1, the signal 34_1 is selected and when the signal on the signal line 33 is 0, the signal line 34_0 is selected. FIG. 4P shows a master-slave D flip-flop, which has at least an input signal line 36 connected to its data terminal D, an input signal line 37 connected to its clock terminal and an output signal line 38 connected to its Q terminal. In some cases, the flip-flop is further provided with an output signal line 39 connected to its $\overline{Q}$ terminal, a clear signal line 40 and a preset signal line 41. Upon application of a signal "1" from the clear signal line 40, an output signal on the output signal line 38 becomes "0", and when a signal "1" is applied from the preset signal line 41, the output signal on the output signal line 38 becomes "1". This flip-flop reads therein data on the line 36 upon rising of a clock signal on the line 37. FIG. 4Q shows a trigger flip-flop, which has a trigger input signal line 42, a clear signal input signal line 40 and an output signal line 38 connected to its Q terminal, and the sign of the output Q is inverted upon rising of the trigger input to the flip-flop. FIG. 4R shows another symbol of the master-slave D flip-flop of FIG. 4P. This is used when the flip-flop is employed as a one-clock delay circuit. FIG. 4S shows a counter, which has a clear signal line 43, an input signal line 44 for pulses to be counted and an output signal line 45 on which a signal "1" is retained after counting a 513th input pulse. The numeral "512" of CNT512 means that this counter counts pulses 512 times and the 513th pulse causes to output "1". FIG. 4T is explanatory of the operation of the counter shown in FIG. 4S. After supplied with a clear signal at a moment 46, the counter CNT512 counts pulses 512 times and, upon detection of the 513th pulse, its output signal becomes "1" at that moment 47. As the counter, there are those which count pulses 128 times, six times and twice, respectively. These counters are indicated by CNT128, CNT6 and CNT2 in the same manner as in the case of FIG. 4S.

FIG. 4U shows en bloc a (a=1, 2, . . .) ANDs as illustrated in FIG. 4V. FIG. 4W shows en bloc a (a=1, 2, . . .) ORs as depicted in FIG. 4X. FIG. 4Y shows en bloc a (a=1, 2, . . .) NOTs as depicted in FIG. 4Z. FIG. 5A is identical with FIG. 5B, in which input and output lines are directly connected. FIG. 5L shows that a b-bit input is shifted up by a bits (where b>a) as shown in FIG. 5D. FIG. 5E shows that a b-bit input is shifted down by a bits and outputted as (b−a) bits (where b>a) as depicted in FIG. 5F. FIG. 5G shows that an a-bit input is outputted with a zero added to its high-order side as depicted in FIG. 5H. FIG. 5I shows that high-order 10 bits of a 38-bit input are outputted as they are, and that the low-order 28 bits are divided into two by steps of 14 bits and four bits between high-order two bits and low-order eight bits of each group are outputted together with the abovesaid high-order 10 bits, as illustrated in FIG. 5J. FIG. 5K shows that high-order four-bits of a 64-bit input are removed therefrom and four bits are added to the low-order side thereof to obtain a 64-bit output as illustrated in FIG. 5L.

FIG. 5M shows that the name of a signal on a signal line 55 is D-SIG. FIG. 5N shows that 12 kinds of control signals are present on the signal line 55, and their names are CT1 to CT12. FIG. 5P shows that five signals are provided on the signal line 55, and that their names are CLOCK, e-in, n-in, START and C-out, respectively. FIG. 5Q shows that the number of signals on a signal line 56 is 12, that they are named CT1 to CT12, respectively, and that they are branched into two signals CT2 and CT1 on a signal line 57, . . . three signals CT5, CT11 and CT12 on a signal line 58, and so forth.

The signal value on the signal line is indicated by binary logic "0" or "1", or a binary integer represented as a 2's complement.

GENERAL ARRANGEMENT OF EMBODIMENT

FIG. 6 illustrates the general arrangement of an embodiment of the present invention, in which the parts corresponding to those in FIG. 3 are identified by the same reference numerals and characters. The quotient calculating unit 9 is divided into a quotient calculating pre-processing section 60 and a quotient calculating post-processing section 61, and these processing sections 60 and 61 are interconnected via a signal line 62. The sliced sections 25_1 to 25_8 are provided with input signal lines 63_1 to 63_8, 65_1 to 65_8 and 67_1 to 67_8 and output signal lines 64_1 to 64_8, respectively. The input signal lines 67_1 to 67_7 are grounded and input a signal value "0", and the input signal line 67_8 inputs a signal value "1". The signal value "1" on the signal line 67_8 means that the sliced section 25_8 is the remotest from the quotient calculating unit 9 and on the side of the least significant digit among the sliced sections. Supplied with the signal value "1", one part of the sliced section 25_8 performs a special operation different from operations of the sliced sections 25_1 to 25_7. This will be described later. Reference numerals 8_1 to 8_8 designate controllers 8 in the individual sliced sections 25_1 to 25_8.

Following the principle of the present invention, the cryptosystem of FIG. 6 is supplied with the variables e, n and M from the input signal lines 28_1, 28_2 and 28_3 and performs the operation $C \equiv M^e \bmod n$ to provide the variable C on the output signal line 29. Similarly the variables d, n and C are applied from the input signal lines 28_1, 28_2 and 28_3 to the cryptosystem when implementing the operation $m \equiv C^d \bmod n$, providing the variable M on the output signal line 29.

The cryptosystem receives an operation control signal from the input signal line 63_1 and the controller 8_1 generates a control signal for the entire cryptosystem. The controllers 8_2 to 8_8 do not operate. In other words, the sliced sections 25_1 to 25_8 are made identical in construction and one of the controllers is used. Therefore, instead of providing a controller in each sliced section, a single controller may be separately provided from the sliced sections as is the case of the quotient calculating unit 9.

The operative state of the cryptosystem is reported to the outside via the output signal line 64_1. Various control signals necessary for calculations for cryptography are produced not only by the controller 8_1 but also by the quotient calculation post-processing section 61 and other parts in the sliced section 25_1 than the controller 8_1. The names of signals on the exponentiation control signal line 26, the multiplication control signal line 21 and the division control signal line 22 are EXP-SEL, M-SIG and D-SIG, respectively. The signal line 27 includes 12 lines and their names are CT1 to CT12, respectively.

QUOTIENT CALCULATION PRE-PROCESSING SECTION

FIG. 7 illustrates the quotient calculation pre-processing section, which is formed by a read only memory (ROM) 68. ROM 68 is used instead of the operation of Eq. (15). When a value $[n \cdot 2^{-504}]$ is provided as an address on the signal line 19, ROM 68 provides on the signal line 62 a value $[2^{13} \div [n \cdot 2^{-504}]]$ precalculated and stored therein. With such an arrangement, the value of v calculated by Eq. (15) can be obtained on the signal line 62 by applying high-order bits of the variable n.

QUOTIENT CALCULATION POST-PROCESSING SECTION

FIG. 8 illustrates the general arrangement of the quotient calculation post-processing section 61, which performs the operations of Eqs. (18) and (19). The signal M-SIG on the multiplication control signal line 21 is composed of four signals, each having a signal value $\delta_{4(j-1)+i} 2^j$ (i=0,1,2,3). Incidentally,

$$\sum_{i=0}^{3} \delta_{4(j-1)+i} \cdot 2^j = M_{2j}$$

is apparent from Eq. (14). From the input signal line 24 is applied a value of high-order 11 bits of M_1, obtained by discarding low-order 501 bits of M_1 represented by 512 bits in Eq. (18); from the input signal line 23 is applied a binary signal value of 14 bits obtained by discarding low-order 500 bits of $R_{j+1,j}$ (i=0,1) represented by 514 bits in Eq. (18). An AND element group 70 performs ANDing of M_1 and $\delta_{4(j-1)+i} 2^j \cdot 2^{-504}$ (i=0,1,2,3,4) in Eq. (18); a logic circuit 71 produces the constant 38 in Eq. (18); and a carry save adder (CSA-Q1) performs the addition in Eq. (18) to calculate the value of X_j'. A carry save adder (CSA-Q2) 72 has seven inputs and two outputs, all of which are binary integers of 14-bit width. An AND element group 73, performs the AND-operation necessary for the calculation of $X_j'' \times v$ in Eq. (19). That is, the AND element group 73_1 is supplied with the value v of six-bit width from the signal line 62 and the value X_j'' from an adder 72 and performs ANDing of each digit of v represented as a binary number and each digit of X_j'' represented as a binary number.

The results of the ANDing are added by a 12-input, 2-output carry save adder (CSA-Q2) 73_2 to obtain the value $X_j'' \times v$. Each output from the adder 73_2 is applied to a circuit 73_3 in which 13 bits are discarded from it, and a value $[X_j'' \times v \times 2^{13}]$ is obtained as the sum of signals which are provided on signal lines 73_4 and 73_5. The signals on the signal lines 73_4 and 73_5 are respectively added in one-output carry propagation adders 74_1 and 74_3, and the signals on the signal lines 73_4 and 73_5 and -1 are added together in three-input, two-output carry save adder (CSA-Q3) 76. The addition results are added in the carry propagation adder 74_2.

On an output signal line 78_1 of the adder 74_1 is provided a value $[X_j'' \times v \times 2^{-13}]+1$. While on an output signal line 78_2 of the adder 74_2 is provided a value $[X_j'' \times v \times 2^{-13}]-1$. The signal on the signal line 78_2 is inverted, providing on a signal line 78_3 the binary value $[X_j'' \times v \times 2^{-13}]-1$ with its respective bits inverted, that is, the absolute value of $[X_j'' \times v \times 2^{-13}]$, i.e.

$|[X_j'' \times v \times 2^{-13}]|$. On a most significant bit output signal line 78_4 of the adder 74_3 is obtained a value "0" or "1" depending on whether the sign of $[X_j'' \times v \times 2^{-13}]$, i.e., the sign of X_j'' is $X_j'' \geqq 0$ or $X_j'' < 0$. The AND of the inverted signal of the signal on the signal line 78_4 and the signal on the signal line 78_1 is obtained in the form of $[X_j'' \times v \times 2^{-13}]$ on a signal line 79_1 when $X_j'' \geqq 0$. The AND of the signals on the signal lines 78_4 and 78_3 is provided as $|[X_j'' \times v \times 2^{-13}]|$ on a signal line 79_2 when $X_j'' < 0$. The signal on the signal line 78_1 is applied to a 32-detector 75_1, which provides a value +31 on a signal line 79_3 when $X_j'' \geqq 0$ and $[X_j'' \times v \times 2^{-13}] + 1 = 32$. The inverted signal on the signal line 78_2 is supplied to a 32 detector 75_2 to provide a value $|-31|$ on a signal line 79_4 when $X_j'' < 0$ and $|[X_j'' \times v \times 2^{-13}]| = 32$. Since the range of Q_j'' is $-31 \leqq Q_j'' \leqq 31$, $|Q_j''|$ can be represented by five bits. The OR of the corresponding bits of the 5-bit signals on the signal lines 79_1 to 79_4 is provided on a signal line 80. The signal on the signal line 80 is composed of five bits of $|Q_j''|$ of Q_j'' defined in Eq. (19). On the signal line 78_4 is provided a sign q_s of Q_j'' which is "0" or "1" depending on whether $X_j'' \geqq 0$, i.e. $Q_j'' \geqq 0$, or $X_j'' < 0$, i.e. $Q_j'' < 0$. On a signal line 82 which is a combination of the signal lines 80 and 78_4 there are provided the most significant bit in the form of q_s and the other five bits in the form of $|Q_j''|$.

On the division control signal line 22 there are provided by the operation of as selector 83 the content of the signal line 82 when CT10=0 and "100001", i.e. −1, from a circuit 75_3 when CT10=1.

For performing the operations of Eqs. (18) and (19), the quotient calculation post-processing section 61 is supplied with high-order 14×2 bits of $R_{j+1,j}$ (i=0,1) from the signal line 23, high-order 11 bits of M_1 from the signal line 24 and four bits of $\delta_{4(j+1)+i}2^i$ (i=0,1,2,3) from the signal line 21. The quotient calculation post-processing section 61 calculates X_j'' in accordance with Eq. (18) and calculates Q_j'' by Eq. (19) in accordance with the condition whether the next $X_j'' \geqq 0$ or $X_j'' < 0$. When CT10=0, the absolute value $|Q_j''|$ of Q_j'' is represented by five bits and the sign of Q_j'' is represented by one bit; namely, a total of six bits is provided on the division control signal line 22. In this case, however, the sign q_s of Q_j'' is represented by 0 or 1 depending on whether $Q_j'' \geqq 0$ or $Q_j'' < 0$. When CT10=1, the absolute value of Q_j'' is 1 and the sign q_s of Q_j'' is 1.

DETAILS OF QUOTIENT CALCULATION POST-PROCESSING SECTION

FIG. 9 illustrates a specific example of the AND element group 70, in which $\delta_{4(j-1)+i}2^i \cdot 2^{-504}$ (i=0,1,2,3) from the signal line 21 and M_1 of eleven bits from the signal line 24 are ANDed with each other, thereby to perform the operation $M_1 \cdot \delta_{4(j-1)+1} \cdot 2^i \cdot 2^{-504}$ in Eq. (18).

FIG. 10 illustrates a logic circuit 71 for producing the constant S=38 in Eq. (18). FIG. 11 shows a seven-input two-output carry save adder (CSA-Q1) 72, which is constituted by a combination of three-input two-output carry save adders (CSAUQ1) 90_1 to 90_5. Each of the three-input two-output carry save adders (CSAUQ1) 90_1 to 90_5 is arranged so that corresponding bits of the three inputs are respectively added by full adders of the same number as bits of each input, as shown in FIG. 12. FIG. 13 illustrates a 12-input two-output carry save adder (CSA-Q2) 73_2, which is made up of three-input two-output carry save adders (CSAUQ2) 91_1 to 91_{10}. FIG. 14 illustrates, by way of example, one of the two-input one-output carry propagation adders 74_1 to 74_3, which is arranged so that corresponding bits of the two inputs are respectively added by full adders of the same number as bits of each input, and carry of each full adder is provided in ascending order.

Sliced Sections

FIG. 15 illustrates, by way of example, the arrangement of one of the sliced sections 25_1 to 25_8 in FIG. 6 in which there are provided registers 101, 102, 103, 104 and 105, each corresponding to one of the eight parts into which the M, e, n, C and M_2 registers 1, 2, 3, 4 and 5 are each divided. To the least significant ends of the registers 101 to 105 are respectively connected input signal lines 101_R to 105_R for supplying thereto signals from a lower-order sliced section. To the most significant ends of the registers 101 to 105 are connected output signal lines 101_L to 105_L for supplying therefrom signals to a higher-order sliced section. A selector 106, one of eight parts into which the selector 6 is divided, is provided, which is controlled by a signal on an input signal line 113. A main adder 110, one of eight parts into which the main adding section 10 for mainly performing the additions in Eqs. (20) and (23) is divided, is provided. Connected to the main adder 110 are input signal lines 114 and 115 and an output signal line 116. The content of the register 103 and a signal on the input signal line 103_R are provided via a signal line 117 to the main adder 110. The content of the most significant bit of the register 102 is applied via the signal line 18 to the controller 8.

Signals for controlling the operation of the sliced section are provided via the five input signal lines 63, and their signal names are CLOCK, e-in, n-in, START and C-out. The operative state of the sliced section is reported to the outside thereof via the three signal lines 64, and their signal names are CT2, n-end and CRYPT-end. A signal indicating the state of carry propagation of each of the plurality of sliced sections is applied via the input signal line 65, and a signal indicating the state of carry propagation in the main adder 110 is provided via the output signal line 66 to the outside of the sliced section, this signal name being CRY-end. A signal indicating that the sliced section 25 is the remotest from the quotient calculating unit 9 like the sliced section 25_8 in FIG. 6, is provided via the signal line 67 and the name of this signal is TAIL. When the signal TAIL is "1", the sliced section 25 is the remotest from the quotient calculating unit 9. Following the exponentiation procedure the sliced section 25 executes Eqs. (16), (17) and (20) to (24) on the premise of Eq. (14). Eq. (15) is executed by the quotient calculation pre-processing section 60 and Eqs. (18) and (19) are executed by the quotient calculation post-processing section 61. In the case where the quotient calculating unit 9 and a plurality of sliced sections are connected as shown in FIG. 6, main signals of each sliced section and the calculation for cryptography bear such relationships as described below. Details of the signals will be described later.

The cryptosystem applies the variable e to the plurality of registers 102 (hereinafter referred to as the e-registers) of the plurality of sliced sections 25 upon application of the signal e-in from the control input signal line 63, applied the variable n to the plurality of registers 103 upon application of the signal n-in, and applies the variable M to the plurality of registers 101 upon application of the signal START. After application of the variable M, the e-registers 102 continue bit-by-bit circular left

shifting until the most significant digit (MSD) of each e-register 102 becomes "1".

Next, upon application of the signal CT5, the cryptosystem performs the operation of Step 1 of the exponentiation procedure;

That is, the operation C←1 is executed.

Next, upon application of the signal CT6, the operation M_2←C in step 2a or M_2←M in step 2b of the exponentiation procedure is executed. (Here, M_1←C always holds on account of the arrangement of the cryptosystem.). Next, in the period in which the signal CT7 becomes "1", the multiplication and division $R=M_1\times M_2 \bmod n$ in step 2a or 2b of the exponentiation procedure are executed and, upon application of a signal MDEND, the multiplication and division are finished. Then, C←R is established owing to the arrangement of the cryptosystem.

The execution of the multiplication and division $R=M_1\times M_2 \bmod n$ based on the exponentiation procedure is controlled as follows: The value of the signal EXP-SEL is determined by each bit e_j of the variable e. When the signal EXP-SEL is "0", step 2a of the exponentiation procedure is executed, and when the signal EXP-SEL is "1", step 2b of the exponentiation procedure is executed. Upon completion of the operation of Eq. (1), i.e. $C=M^e \bmod n$, by the above calculation, the value of the signal CRYPT-end is altered from "0" to "1" and, upon application of the signal C-out, the variable C obtained by the calculation for cryptography is outputted.

With such an arrangement, the calculation for cryptography can be achieved following the principle of the present invention by connecting the quotient calculating unit 9 and the plurality of sliced sections as shown in FIG. 6. The same is true of the case where the quotient calculating unit 9 is divided into the quotient calculation pre-processing section 60 and the quotient calculation post-processing section 61.

Details of Sliced Sections

The registers 101, 103, 104 and 105 are formed as four-bit parallel input-output shift registers, as shown in FIGS. 16, 17, 18 and 19, respectively, and they are shifted by signals CT4, CT3, CT12, and CT6 and CT9, respectively. The register 104 is capable of presetting in parallel a 64-bit signal from a signal line 116 under the control of the signal CT11. In the case where the signal TAIL is "1" when the signal CT5 is provided, "1" is preset only in the least significant bit of the register 104 and other bits are preset to "0", and where the signal TAIL is "0", the register 104 is entirely cleared by the application of CT5. The register 105 is also controlled by the signal CT6 and capable of presetting the 64-bit signal M_2 in parallel. The register 102 is constituted as a one-bit shift register as shown in FIG. 20 and it is shifted by the signal CT1. In the sliced section 25_8, when the signal CT2 becomes "1", the register 102 is put in its circular operation. FIG. 21 illustrates a specific example of the selector 106.

FIG. 22 illustrates the general arrangement of an embodiment of the main adder 110. An $M_1 \cdot M_{2j}$ calculator 140 for calculating $M_1 \cdot M_{2j}$ seen in FIG. 22 is arranged as depicted in FIG. 23. A $-Q_j n$ calculator 150 for operating $-Q_j n$ is arranged as shown in FIG. 24. By the sign bit of the signal Q_j on a division control signal line 134 is controlled a selector (SEL-Q) 151 to select a signal n on a signal line 152 from the n-register 103 and a signal line 154 from the next lower-order sliced section and a signal $\bar{n}$ on a signal line 153 from the n register 103 and a signal line 155 from the next lower-order sliced section. And the selected signal and the signal Q_j on the signal line 134 are ANDed. An adder 160 seen in FIG. 22 is formed by three-input two-output carry save adders 161_1 to 161_{10} as shown in FIG. 25. As shown in FIG. 26, the three-input two-output carry save adder 161 has 66 bits for each input and output, and the most significant one of 64 bits on the lower-order side in the adder 161 is branched to be applied to the corresponding carry save adder 161 of the next higher-order sliced section as indicated by a signal line 880. A signal applied via a signal line 880' from the corresponding lower-order side is provided to the side of the carry outputs from all the full adders FA. Circuits 170_L and 170_R in FIG. 22 are 66-bit registers as shown in FIG. 27. A circuit 180 in FIG. 22 adds two outputs from the adder 160 of this sliced section by a carry propagation adder 184 to produce output as shown in FIG. 28. Carries resulting from this addition are applied to the next higher-order sliced section one after another. In the most significant sliced section 25_1, carry components in the output from the adder 160 are added by an adder 186 and a portion of the addition result is supplied to the controller 8_1 via a signal line 187. A carry detector 190 in FIG. 22 performs ORing of NOT outputs of the exclusive ORs of corresponding bits of two added outputs from the adder 160 as shown in FIG. 29, and the detector 190 yields an output "0" or "1" depending on whether a carry to be transferred to the higher order is produced from the addition of the 66 bits in the adder 160.

In FIG. 22, selectors 301 and 302 are controlled by the signal CT10 to select a signal obtained by multiplying each calculation result of the corresponding registers 170_L and 170_R by 2^4 and a signal corresponding directly to the calculation result. That is, in the case of the compensating calculation, the signal corresponding to the calculation result is selected and, when the signal obtained by the multiplication, high-order four bits from the next lower order sliced section are added to less significant side of the selected signal.

FIG. 30 shows the state in which the registers 101_1 to 101_8 of the sliced sections 25_1 to 25_8 shown in FIG. 6 are coupled together to form the register 1 of 512-bit length because $64\times 8=512$. The register 1 stores the variable M of 512-bit length. FIG. 31 illustrates the state in which the registers 102_1 to 102_8 of the sliced sections 25_1 to 25_8 are coupled together to set up the e-register 2 of 512-bit length, which stores the variable e of 512-bit length. The e-register 2 has the function of circularly shifting signals of 512 bits to left bit by bit. FIG. 32 illustrates the state in which the registers 103_1 to 103_8 of the sliced sections 25_1 to 25_8 are coupled together to constitute the register 3 of 512-bit length, which stores the variable n of 512-bit length. FIG. 33 shows the state in which the registers 104_1 to 104_8 of the sliced sections 25_1 to 25_8 are coupled together to form the C-register 4 of 512-bit length, which stores the variable R(C) of 512-bit length. FIG. 34 shows the state in which the registers 105_1 to 105_8 of the sliced sections 25_1 to 25_8 are coupled together to form the M_2-register 5 of 512-bit length, which stores the variable M_2 of 512-bit length. FIG. 35 shows the state in which the selectors 106_1 to 106_8 of the sliced sections 25_1 to 25_8 are coupled together to serve as the selector 6 of two inputs and 512-bit width.

FIG. 36 illustrates the state in which the main adders 110_1 to 110_8 of the sliced sections 25_1 to 25_8 are coupled together to form the main adder 10 of 514-bit width. FIG. 37 shows the state in which the $M_1 \cdot M_{2j}$ calculators 140_1 to 140_8 of each main adder 110 of the sliced sections 25_1 to 25_8 are coupled together and the input signal line 114_a (a=1, 2, . . . 8) are divided into input signal lines 114_{La} and 114_{Ra}. Because of such coupling, ANDing of $M_1 \cdot M_{2j}$ (where M_1 is 512-bit and M_{2j} is four-bit) in Eq. (20) can be performed. FIG. 38 shows the coupling state of the $-Q_j \cdot n$ calculators 150_1 to 150_8 of each main adder 110 of the sliced sections 25_1 to 25_8, by which ANDing of $-Q_j'' \cdot n$ in Eq. (20) can be carried out. FIG. 39 shows the coupling state of adders 160_1 to 160_8 of each main adder 110 of the sliced sections 25_1 to 25_8. FIG. 40 shows the coupling state of the registers 170_{L1} to 170_{L8} of each main adder 110 of the sliced sections 25_1 to 25_8. Also the registers 170_{R1} to 170_{R8} are similarly coupled. FIG. 41 shows the coupling state of the circuits 180_1 to 180_8 of each main 110 of the sliced sections 25_1 to 25_8. FIG. 42 shows the coupling state of the carry detectors 190_1 to 190_8 of each main adder 110 of the sliced sections 25_1 to 25_8 with the circuit 135_1 of the sliced section 25_1.

FIG. 43 is explanatory of operations in FIGS. 39 to 40. The circuits 160, 170_L, and 170_R and 180 each perform a 66-bit calculation in the sliced sections 25_1 to 25_8 but, in the coupled state, the sliced sections 25_2 to 25_8 each perform a 64-bit calculation. Thus a calculation of a total of 512+2=514 bits is conducted. FIGS. 44 and 45 illustrate the coupling operation of the $M_1 \cdot M_{2j}$ calculator 140 in FIG. 37.

From input signal lines 114_{L1} to 114_{L8} in FIG. 37 are applied to the sliced sections 25_1 to 25_8 the variable M_1 by steps of 64 bits, from each of signal lines 114_{R1} to 114_{R7} are applied high-order three bits of the input on each of the signal lines 114_{L2} to 114_{L8}, and from a signal line 114_{R8} is applied a signal "0" of three bits. As a result of this, the ANDing of $M_1 \cdot M_{2j}$ (M_1 being 512-bit and M_{2j} 4-bit) can be achieved. The number of significant digits used for the operation $M_1 \cdot M_{2j}$ is 514 from the low-order end, and 515th and higher-order bits are neglected but this does not matter for the reasons already described.

FIG. 46 shows the coupling operation of a $-Q_j \cdot n$ calculator shown in FIG. 38 (also see FIG. 24). Signal lines 152_1 to 152_8 equally divides n (512 bits) into eight by steps of 64 bits, and apply them to the $-Q_j \cdot n$ calculator from the side of the high-order position. Signal lines 153_1 to 153_8 equally divide inverted signals of the respective bits of n into eight by steps of 64 bits and apply then from the side of the high-order position. Signal lines 154_1 to 154_7 apply high-order four bits of the signals on the signal lines 152_2 to 152_8, respectively. A signal line 154_8 applies a signal "0000". Signal lines 155_1 to 155_7 apply high-order four bits of the signals on the signal lines 153_2 to 153_8. A signal line 155_8 applies a signal "0000" when the signal TAIL from a signal line 156 (see FIG. 24) is "1". As a result of this, the ANDing of $-Q_j''$ and n can be performed. The number of significant digits for the operation $-Q_j'' \times n$ is 514 from the low-order end, and 515th and higher-order bits are neglected but this does not matter for the reasons already given.

FIG. 47 is explanatory of the coupling operation of the register 170 shown in FIG. 40. The registers 170_{L1} to 170_{L8} serve as a 514-bit register as a whole in the same manner as described previously in respect of FIG. 44. When the signal CT10 is "1", signals of the registers 170_{L1} to 170_{L8} are provided, as they are, on signal lines 171_{L1} to 171_{L8}. When the signal CT10 is "0", signals resulting from shifting of the registers 170_{L1} to 170_{L8} to the high-order side by four bits are provided on the output signal lines 171_{L1} to 171_{L8}. As a result of this, since values of $R_{j+1,1}$ and $R_{j+1,0}$ are stored in the registers 170_L and 170_R, respectively, $2^4 \cdot R_{j+1,1}$ and $2^4 \cdot R_{j+1,0}$ are provided on the signal lines 171_L and 171_R, respectively, when the signal CT10 is "0" and, when the signal CT10 is "1", $R_{j+1,1}$ and $R_{j+1,0}$ are provided on the signal lines 171_L and 171_R. The condition CT10=0 permits the addition in Eq. (20) and the condition CT=1 permits the addition in Eq. (23).

FIG. 48 is explanatory of the coupling operation of a carry detector 190 shown in FIG. 42. Arrows 191_1 to 191_8 indicate signal values on output signal lines 66_1 to 66_8 of the carry detectors 190_1 to 190_8, respectively.

CONTROLLER

FIG. 49 shows the general arrangement of the controller 8, which comprises first to fifth controllers (CTL1), 230, (CTL2) 250, (CTL3) 260, (CTL4) 270, (CTL5) 280 and other related circuits. From an input signal line 203 are applied a signal CLOCK to all the controllers 230 to 280, the signals e-in, n-in and START to the first controller 230 and the signal C-out to the fifth controller 280. From an input signal line 205 is applied a signal CARRYEND to the fourth controller 270, and from an input signal line 206 is applied a signal SIGN to the fourth controller 270. On an output signal line 204 are provided signals CT2 and n-end from the first controller 230 and the signal CRYPT-end from the second controller 250. On an output signal line 220 of the fourth controller 270 is provided therefrom the signal CT10. On an output signal line 221 of the third controller 260 is provided therefrom the signal EXP-SEL. On an output signal line 227 connected to all the controllers are provided thereon the signals CT1 to CT12. An output signal line 251 of the second controller 250 transmits a signal SFT1 to an OR circuit 800, an output signal line 252 transmits the signal CT5 to the third controller 260 and the signal line 227, and an output signal line 253 transmits a signal es-end to the third controller 260. The third controller 260 applies the signal CT7 via an output signal line 263 to the fourth controller 270 and the signal line 227. The fourth controller 270 applies a signal MDEND via an output signal line 264 to the third controller 260 and a delay circuit 801. From the signal line 18 is supplied e_i in the variable e to the second controller 250.

FIGS. $50A_1$ to $50U_1$ and correspondingly continued FIGS. $50A_2$ to $50U_2$ show waveforms of the signals CLOCK, e-in, CT1, CT2, n-in, CT3, n-end, START, CT4, MDEND, CT5, SFT1, es-end, CT6, CT7, MDEND, e-out, CT11, CT12 and CRYPT-end which occur at respective parts of the controller of FIG. 49 while in operation.

Next, a description will be given, with reference to FIG. 50, of the operation of the controller 8 shown in FIG. 49. The controller 8 inputs thereinto and outputs therefrom signals for controlling the operation $C \equiv M^e \bmod n$ in the following manner: The signal CLOCK of the cryptosystem is always applied to the controller 8. Upon application of the signal e-in at a moment t_1, the first controller 230 outputs therefrom the variable e input command signal CT1, by which the variable e is input bit by bit by 512 clocks. Upon completion of this,

the first controller 230 outputs, at that moment t_2, the signal CT2 representing the completion of the input of the variable e.

Next, upon application of the signal n-in at a moment t_3, the first controller 230 yields the variable n input command signal CT3, inputting the variable n by steps of four bits by 128 clocks. Upon completion of this, the first controller 230 yields the signal n-end representing the completion of the input of the variable n at that moment t_4.

Next, when the signal START is applied at a moment t_5, the first controller 230 outputs a variable M input command signal CT4 commanding to input the variable M by steps of four bits by 128 clocks. Upon completion of the input of the variable M, the controller 230 yields the signal MDEND representing the end of the input of the variable M at that moment t_6. At the same time, the controller 230 yields the signal CT5 for initializing the registers (FIG. 15) within the cryptosystem prior to starting the operation $C \equiv M^e \bmod n$.

Next, the second controller 250 generates the signal SFT1 by which the content of the e-register 102 having stored therein the variable e is circularly shifted to left bit by bit, and outputs this signal as the signal CT1 via the OR circuit 800 starting at a moment t_7. At this time, the signal CT1 is provided as clock pulses of the same number as the number of 0s on the higher-order side of the variable e represented by 512 bits. When the most significant bit (MSB) of the 512-bit-wide e-register having stored therein the variable e becomes "1" after repeating such circular left shifting bit by bit, the second controller 250 yields the signal es-end representing completion of the signal SFT1 at a moment t_8. Then, the following various signals are produced for executing the steps 2a and 2b of the exponentiation procedure.

Upon outputting the signal es-end, the third controller 260 generates the signal CT6 for preparing the start of the operation for the multiplication-division $R \equiv M_1 \times M_2 \bmod n$ first and then yields the signal CT7 indicating the operation. By this, all the main adders 110_1 to 110_8 of the sliced sections 25_1 to 25_8 respectively execute the multiplication-division $R \equiv M_1 \times M_2 \bmod n$. Upon reception of the signal MDEND indicating the completion of this multiplication-division at a moment t_9, the signal CT7 from the third controller 260 is made a 0. The signal CARRYEND on the signal line 205 and the signal SIGN on the signal line 206 are utilized during execution of the multiplication-division. This will be described later in detail. Upon each completion of the multiplication-division, the signals CT6 and SFT1 are output to repeatedly perform the operation $C \equiv M_1 \times M_2 \bmod n$. But when e_i of the variable e shifted into the most significant bit (MSB) of the e-register is "1" immediately after the execution of the step 2a of the exponentiation procedure, the signal SFT1 is "0". The signal CT7 is output as a signal indicating the periods of execution of the steps 2a and 2b of the exponentiation procedure. During execution of the multiplication-division, the signal EXP-SEL commanding switching of the selectors 106_1 to 106_8 is provided on the signal line 221. Here, when the value of the signal EXP-SEL is 0, the step 2a of the exponentiation procedure is executed and, when the signal EX-SEL is 1, the step 2b is executed. Upon completion of the exponentiation, the signal CRYPT-end is derived from the second controller 250.

Upon inputting the signal C-out commanding to bring out the variable C from the cryptosystem at a moment t_{10}, the fifth controller 280 outputs the signal CT12 instructing that the variable C be output by steps of four bits by 128 clocks, and the signal CT11 representing the period for which the signal CT12 is valid remains at 1 during the above operation.

In this way, the controller 8 inputs therein and outputs therefrom signals for controlling a series of calculations for inputting the variables e, n and M, executing the operation $C \equiv M^e \bmod n$ and outputting the variable C.

The following will describe details of operations of the signals CARRYEND and SIGN and specific arrangements of the controllers 230, 250, 260, 270 and 280.

FIG. 51 illustrates a specific example of the first controller (CTL1) 230 and FIGS. 52A to 52J show the waveforms of signals which occur at respective parts of the first controller 230 while in operation, the waveforms being labeled with corresponding signal names on the left-hand side.

When the signal e-in from a signal line 231 is input via a delay circuit 805 to a flip-flop 806, the output from the flip-flop 806 goes to a 1 to open a gate 807. Then the signal CLOCK on the signal line 240 is applied via the gate 807 to a counter 808 for counting and, at the same time, it is applied to a gate 809 to output therefrom a signal CT1' on an output signal line 234. The signal CT1' is provided to the OR circuit 800 in FIG. 49, producing the signal CT1. When the count content of the counter 808 reaches 512, the gate 809 is closed. That is, 512 signals of CT1' are generated. Further, the output from the counter 808 is sent as the signal CT2 on a signal line 238. When the signal n-in is provided on a signal line 232, the signal CT3 is output from a signal line 235 by 128 clocks, after which the signal n-end is sent on a signal line 239. When the signals CT2 and n-end are both being generated, a gate 814 is opened. Next, when the signal START is applied to the gate 814 from a signal line 233, the signal CT4 is similarly output 128 times on a signal line 236 in synchronism with clocks by means of a flip-flop 815, gates 816 and 818 and a counter 817, after which the signal MEND is sent on a signal line 237. In this way, the first controller 230 controls inputting of the variables e, n and M.

FIG. 53 illustrates a specific example of the second controller (CTL2) 250 and FIGS. 54A to 54G show signal waveforms which occur at respective parts of the second controller 250 while in operation. When the signal MEND is applied via the signal line 237 from the first controller 230, the signal CT5 is provided on a signal line 252 from a gate 820 for the delay time of a delay circuit 819. Further, while the signal MEND is applied and the signal e_i from a signal line 256 remains at a 0, gates 821 and 822 are opened to permit the passage therethrough of the signal CLOCK, which is provided as the signal SFT1 on a signal line 251 via an OR circuit 823. By the signal SFT1 the e-register 102 in FIG. 15 is shifted to left. When the most significant bit of the e-register 102 of the sliced section 25_1 goes to a 1, the signal e_i from the signal line 256 also goes to a 1 to cause a Q output of a flip-flop 824 to go to a 1, opening a gate 825 and outputting the signal es-end via a gate 826 on a signal line 253. Thereafter, upon each application of the signal SFT2 from a signal line 254, it is output as the signal SFT1 via the gate 825 and the OR circuit 823. The outputs from the OR circuit 823, that is, the signals SFT1 are counted by a counter 827, which provides the signal CRYPT-end on a signal line 255 when having counted 512 after inputting of the signal CT5.

In this way, when supplied with the signal MEND representing completion of inputting the variable M, the second controller 250 performs control of circularly shifting the content of the e-register to left until its most significant bit goes to a 1, yielding the signal SFT1 for a circular shift of the e-register left one bit position upon each application of the signal SFT2 and outputting the signal CRYPT-end after the circular shift of the e-register left a total of 512 bit positions, i.e. after one circular shift cycle of the e-register.

FIG. 55 illustrates a specific example of the third controller (CT3) 260 in FIG. 49, and FIGS. 56A to 56H show, by way of example, signal waveforms which occur at respective parts of the third controller 260 while in operation.

Upon application of the signal CT5 via the signal line 252 from the second controller 250, flip-flops 828, 829, 830 and 831 are cleared. Upon application of the signal es-end via the signal line 253 from the second controller 250, the signal CT6 is provided via an OR circuit 832 on a signal line 261 and the flip-flop 831 is triggered via an OR circuit 833, providing a Q output of the flip-flop 831 as the signal CT7 on a signal line 263. The operation $R=M_1\times M_2 \bmod n$ is started and, upon completion of this calculation, the signal MDEND is input via a signal line 264 from the fourth controller 270, for example, at a moment t_1. The signal MDEND is applied via the OR circuit 833 to the flip-flop 831 to trigger it, causing the signal CT7 to go from a 1 to a 0. The signal e_i on the signal line 256 and the Q output of the flip-flop 828 are provided to a NOT EXCLUSIVE OR 834, and its output and the signal MDEND are provided to an AND gate 835, so that if the signal e_i is a 1 when the signal MDEND is applied at the moment t_1, the output from the NOT EXCLUSIVE OR 834 is a 0 and the output from the AND gate 835 remains at a 0, resulting in the signal SFT2 being not output on the signal line 254 as shown at a moment t_2. Moreover, since the signal MDEND, the signal e_i on the signal line 256 and an $\overline{Q}$ output of the flip-flop 828 are provided to an AND gate 836, the Q output from the flip-flop 828 goes to a 1 in the case where the signal e_i is at a 1 at the time of application of the signal MDEND. Furthermore, the signal MDEND at the moment t_1 passes through the flip-flops 829 and 830, thereafter being sent as the signal CT6 via a gate 837 and the OR circuit 832 on the signal line 261 at a moment t_3. The output from the flip-flop 830 is provided via a gate 838 and the OR circuit 833 to the flip-flop 831 to trigger it, generating the signal CT7 at a moment t_4. Consequently, the operation $R=M_1\times M_2 \bmod n$ is resumed; namely, the step 2b is executed. When the signal MDEND is applied again at a moment t_5, the same operations as described above are carried out but, in the case where the signal e_i is at a 1, the output from the circuit 834 goes to a 1, yielding the signal SFT2 as shown at the moment t_6. In the case where the signal e_i is at a 0 when the signal MDEND occurs at the moment t_1, however, the output from the circuit 834 goes to a 1 to generate the signal SFT2 and, by the next signal CT7, the step 2a is executed. At this time, the Q output of the flip-flop 828 is made a 0.

Thus, in the exponentiation procedure, if the condition $e_i=0$ holds immediately after the step 2a, then the content of the e-register 102 is shifted one bit position and an operation $i\leftarrow i-1$ is performed; if $e_i=1$ immediately after the completion of the step 2a, then the step 2b is executed and the content of e register 102 is shifted one bit position, thereafter the operation $i\leftarrow i-1$ is executed. These procedures are repeated until i reaches 0. Since the gate 837 is closed when the signal CRYPT-end is applied via the signal line 255 from the second controller 250, even if the signal MDEND is applied, the signal CT6 is not generated as indicated at a moment t_6.

Thus, in the exponentiation procedure, after the calculation of the step 2 has been controlled, that is, after the variable e has been made $e_k, e_{k-1}, \ldots e_1, e_0$ in the binary representation, the steps 2a and 2b are executed in the order $i=k, k-1, \ldots 1, 0$.

FIG. 57 illustrates a specific example of the fourth controller (CTL4) 270 shown in FIG. 49, and FIGS. 58A to 58H show, by way of example, signal waveforms which occur at respective parts of the fourth controller 270 while operation. Upon application of the signal CT7 via the signal line 263 from the third controller 260, the signal CT8 is provided on a signal line 271 from a gate 840. By the signal CT8, a counter 841 and a flip-flop 842 are cleared, and counters 276 and 277 are cleared via an OR circuit 843. By the signal CT7, a gate 844 is opened, through which the signal CLOCK from the signal line 240 is applied to the counter 841 for counting and, at the same time, the signal CT9 is provided via a gate 845 on a signal line 272. When the counter 841 has counted the signal CLOCK up to 128, the gate 845 is closed by the output from the counter 841 to stop sending out of the signal CT9 but, on the other hand, a gate 846 is opened, permitting the counters 276 and 277 to start counting the signal CLOCK at a moment t_1. At the moment t_1 successive calculations of $R_1=M_1\times M_2 \bmod n$ is completed and $R_1\geqq 0$ is checked in Eq. (22). In the case where the signal CARRYEND is at a 0 on a signal line 275 when the counter 226 has counted the signal CLOCK up to two after the moment t_1, a gate 847 remains closed and, at a moment t_2 when the counter 277 has counted the signal CLOCK up to six, the output from the counter 277 is applied via an OR circuit 848 to a gate 849, by the output of which gates 850 and 851 are opened for a fixed period of time. At this moment t_3, the signal SIGN on a signal line 274, that is, the sign of

$$R_1 = \sum_{i=0}^{1} R_{1,i}$$

in Eq. (22), is checked. When the signal SIGN is 1, that is, when $R_1<0$, the signal CT10 is sent on a signal line 275 via the gate 851. By this, the compensating calculation of Eq. (23) is performed. At this time, the counters 276 and 277 are cleared by the output of the gate 849 via the OR circuit 843 but, at a moment t_4 of completion of this clearing, the counters 276 and 277 start counting again. At a moment t_5 when the counter 276 has counted the signal CLOCK up to two, the gate 847 is opened and if the signal CARRYEND on the signal line 275 is 1, the output of the gate 847 is provided via the OR circuit 848 to the gate 849 and, by the output of the gate 849, the gates 850 and 851 are opened at t_6, checking the signal SIGN, that is, the sign of R_1. At this time, when the signal SIGN is at a 1, the signal CT10 is output at the moment t_6. Similarly, the compensating calculation of Eq. (23) is performed and then, at a moment t_7 when the gates 850 and 851 are opened, if the signal SIGN is at a 0, the signal MDEND is provided from the gate 850 on a signal line 264. By this signal MDEND, the signal CT7 is made a 0 as described previously in

respect of FIG. 55. The signal CT7 is a signal that holds a 1 during the operation $R=M_1 \cdot M_2 \bmod n$. Further, a Q output of the flip-flop 842 is caused by the signal MDEND to go to a 1, and a gate 852 is opened, through which the supply of the signal CLOCK to the counters 276 and 277 is continued, preventing occurrence of the signal CT10 while the signal CT7 is at a 0.

In this way, the compensating calculation for the multiplications and divisions in Eqs. (22) to (24) can be controlled. FIG. 59 illustrates a specific example of the fifth controller 280 shown in FIG. 49 and FIGS. 60A to 60D show signal waveforms which occur at respective parts of the controller 280 while in operation. Upon application of the signal CRYPT-end via a signal line 282 from the second controller 250, a gate 853 is opened by the signal CRYPT-end. If the signal C-out is input from a signal line 281 in this state, a flip-flop 854 is driven via the gate 853 and its Q output goes to a 1, which is output as the signal CT11 on a signal line 283 via a gate 855. And, by the output of the flip-flop 854, a gate 856 is opened and the signal CLOCK on the signal line 240 is counted by a counter 857. At the same time, the signal CT12' is provided via a gate 858 on a signal line 284, and output as the signal CT12 via the OR circuit in FIG. 49, and the calculation result in the C-register 104 is output from the cryptosystem. When the counter 857 has counted up to 128, the gates 855 and 858 are both closed, stopping the both signals CT11 and CT12'.

After the operation $C \equiv M^e \bmod n$ has thus been completed, the variable C of 512 bits can be output from the cryptosystem by steps of four bits by 128 clocks.

In the quotient calculation pre-processing section 60 shown in FIG. 7, n ($2^{511}<n<2^{512}$) is input and v is obtained by Eq. (15), i.e. $v \leftarrow [2^{13} \div [n \cdot 2^{-504}]]$. A supplementary description will be given of the size of ROM 68 of the pre-processing section 60. The address of ROM 68 can be represented by a positive integer of eight-bit width based on the condition $2^7<[n \cdot 2^{-504}]<2^8$. Since 2^7 or less addresses are not used, however, the size of ROM 68 may be one-half of that of ROM having 2^7 or less addresses. The value of v can be represented by a positive integer of six-bit width based on the condition $2^5<v<2^6$. But the most significant bit (MSB) of v is always 1 and this value is fixed, so that the value of v except for the "1" of the most significant bit is stored in ROM 68 using five bits and, when to refer to the value of v, one bit having a value "1" is added as the most significant bit of v by an inverter 859. It is also possible, of course, to arrange the pre-processing section 60 so that the ROM itself inputs therein n of eight bits and outputs therefrom v of six bits.

COMPENSATING CALCULATION

The calculations of Eqs. (20) and (21) are repeated and it is checked whether a compensating calculation is required in Eq. (22), and if necessary, the compensating calculation is performed. A description will be given, with reference to FIG. 22, of the compensating calculation. In the period during which the value of the signal CT10 on a signal line 300 holds zero, that is, in the period in which mainly the operations of Eqs. (17) to (22) are performed, input signal lines 303 and 304 of the selectors 301 and 302 are selected and the value of the variable $R_{j+1,l}$ is shifted left four bit positions in each of circuits 861 and 862, selecting value $2^4 \cdot R_{j+1,l}$ necessary for calculating Eq. (20).

When the signal CT10 on the signal line 300 has a value 1, that is, when the compensating calculation of Eq. (22) is executed, input signal line 305 and 306 of the selectors 301 and 302 are selected, that is, the value of the variable $R_{j+1,l}$ is selected. In the quotient calculation post-processing section 61 shown in FIG. 8, the selector 83 selects the output of a circuit 75_3 by the signal CT10 and $Q_j''=-1$ is provided via the signal line 22 to the $-Q_j n$ calculator 150 in FIG. 22. Furthermore, as shown in FIG. 15, an AND gate 136 is supplied with an inverted signal of the signal CT10, and hence is closed, and the output of the M_2-register 105 is not provided on the signal line 105_L. And, the value of the signal on the signal line 105_L, that is, the value $\delta_{4(j-)+i} 2^i$ (i=0, 1, 2, 3) of the signal M-SIG on a signal line 21 in FIG. 6, namely, $M_{2,j}$, becomes 0 and, as a result of this, Eq. (23) is calculated in the adder 160.

This compensating calculation can be changed as follows:

$$\text{Step 6': } R_1 \leftarrow \frac{1}{2} \sum_{i=0} R_{1,l} \tag{22'}$$

$$\left.\begin{array}{l}\text{Step 7': If } R_1 \geqq 0, \text{ then go to step 8'.} \\ \quad R_1 \leftarrow R_1 + n \\ \quad \text{Go back to step 7'.}\end{array}\right] \tag{23'}$$

$$\text{Step 8': } R \leftarrow R_1. \text{ Halt.} \tag{24'}$$

The compensating calculation by Eqs. (22') to (24') can be implemented, for instance, as shown in FIG. 61. The outputs $R_{j+1,1}$ and $R_{j+1,0}$ of the registers 170_L and 170_R are respectively applied via signal lines 313 and 314 to the selectors 311 and 312 at one input thereof and, at the same time, the register outputs are respectively shifted by the circuits 861 and 862 to the left by four bit positions and supplied to the adder 160. High-order 66 lines of the signal line 308 are connected as a signal line 315 to the other input of the selector 311. The signal line 309 is added with high-order two bits and connected as a signal line 316 to the other input of the selector 312. The number of lines of the output signal line 116 is not 64 but increased to 66 and this output is input to the register 104 shown in FIG. 15, from which it is input via the signal line 114 to the main adder 110, so that the input signal line 114 is composed of 66 and three lines.

While the signal CT10 assumes a value 0, that is, while the repetitive calculations Eqs. (7) to (12) and (21') are executed, the selectors 311 and 312 select the signal lines 313 and 314, whereby Eq. (22') is correctly calculated.

When the signal CT10 assumes a value 1, the signal lines 315 and 316 are selected and, in the adder 180, the signals R_1 and n from the selectors 311 and 312 are added. As a result of this, Eq. (23') is correctly computed. The value of the eight-divided signal R_1 on the output signal line 116 of the main adder 110' is provided to the signal line 308 shown in FIG. 61 via the register 104 and the signal line 114 shown in FIG. 15. In other words, the values eight-divided from signals R_1 and n are obtained on the signal lines 315 and 316, respectively, in consequence of which the calculation $R_1 \leftarrow R_1+n$, i.e. Eq. (23') is correctly performed. Here, the value eight-divided from the signal n represents 64 bits obtained by dividing the variable n of 512-bit width equally into eight. The values of the eight-divided sig-

nal R_1 represent eight groups of bits obtained by dividing the 514-bit-width variable R_1 into a group of 66 bits and seven 64-bit groups (i.e. $514=66+64\times7$). In this case, since the adder 160 is not used for the compensating calculation, the circuit 75_3, the selector 83 and the signal line 20 in FIG. 8 are unnecessary, and the signal line 82 is connected directly to the signal line 22. Furthermore, the gate 136 in FIG. 15 is also unnecessary and the four output signal lines of the M_2-register 105 are connected directly to the signal line 105_L. Besides, the C-register 104 in FIG. 15 is made 66-bit-wide, not 64-bit-wide and, in the coupling of the registers 104_1 to 104_8 shown in FIG. 33, the register 104 is constituted as a 514-bit-wide register based on the calculation $512+2=514$ as is the case with FIG. 44.

MODIFICATION OF $-Q_j \cdot n$ CALCULATOR

A description will be given of the main point of another example of the $-Q_j \cdot n$ calculator 150 shown in FIG. 22. $|Q_j''|$ is represented as a binary number

$\sum_{\mu=0}^{2h} H_\mu \cdot 2^\mu$, where h ≧ 3, and it is given by

$$|Q_j''| = Q_{ja} + Q_{jb} + Q_{jc} \quad \text{(E-1)}$$

where

$$Q_{ja} = H_{2h} \cdot 2^{2h} + 2H_{2h-1} \cdot 2^{2h-1} \quad \text{(E-2)}$$

$$Q_{jb} = \sum_{\mu=1}^{h-1} \{-H_{2\mu+1} \cdot 2^{2\mu+1} + H_{2\mu} \cdot 2^{2\mu} + 2H_{2\mu-1} \cdot 2^{2\mu-1}\} \quad \text{(E-3)}$$

$$Q_{jc} = -H_1 \cdot 2^1 + H_0 \quad \text{(E-4)}$$

For instance, in the case where $|Q_j''|=11011$,

$$Q_{ja} = 2^5,\ Q_{jb} = -2^2,\ Q_{jc} = -1$$

With such a representation, $|Q_j''| \cdot n$ requires 5×66 bits if Q_j'' is represented as a mere binary number but, if Q_{ja}, Q_{jb} and Q_{jc} are used, 3×66 bits are sufficient for 2^5, -2^2 and -1, so that 2×66 bits become unnecessary, permitting the reduction of the number of inputs to the carry save adder 160 shown in FIG. 22. FIG. 62 illustrates, by way of example, the circuit arrangement therefor corresponding to that depicted in FIG. 24. In FIG. 62, Q_{ja}, Q_{jb} and Q_{jc} generators 502, 503 and 504 respectively input therein Q_j'' from the signal line 134 and compute Q_{ja}, Q_{jb} and Q_{jc} of the logic shown in FIGS. 63, 64 and 65, thereafter outputting the calculation results. For example, in FIG. 63, a column 2^4 of D-SIG input indicates the digit position of 2^4 of $|Q_j''|$ represented as a binary number. The same is true of 2^3, 2^4, 2^5, -2^4 and -2^5 in the column of output indicate output terminals of the Q_{ja} generator 502 and are caused to have a 1. 0 in the column of output indicates that the signal value at the output terminal is at a 0. For instance, in the case where q_9, 2^4 and 2^3 in a column of input are 0, 1 and 1, it indicates that a signal at the output terminal 2^5 is caused to have a 1. As a result of this, the quantity of data representing $-Q_j'' \cdot n$ is decreased, permitting reduction of the circuit scale of the carry storage type adder 160.

NUMERICAL EXPRESSION OF THE PRINCIPLE OF THE MULTIPLIER-DIVIDER, THE PRINCIPLE PART OF THE CRYPTOSYSTEM OF THE PRESENT INVENTION

The principle of obtaining the quotient Q and the remainder R of a multiplication-division of integers $(M_1\times M_2)\div n$ is shown as a theorem and a system of theorem by numerical expressions.

Theorem

The quotient Q and the remainder R of the multiplication-Division of integers $(M_1\times M_2)\div n$ can be obtained as described hereinafter by Eqs. (F20) to (F22) based on I_j and R_1 obtained by repeating recurrence formulae of Eqs. (F14) to (F19) in an order $j=l, l-1, \ldots, 2, 1$ on the premise of Eqs. (F1) to (F13). Here, Eq. (F17) represents the range over which I_j is obtainable with Eq. (F16), and Eq. (F18) indicates the method of calculation of R_j. In the equations, n, M_1, M_2, R_{j+1} and R_j are variables, m, K, A, λ, ω, S, t_1, and t_2 constants and α_j a random number the value of which irregularly varies as the value of j is changed, t_1 and t_2 being real numbers and the others being integers. Incidentally, since α_j naturally occurs, there is no need of taking it into account when forming adders. This means that α_j may be neglected, for example, in Eq. (F16).

If a multiplication-division $(M_1\times M_2')\div n'$ is performed with $M_2'=M_2\times2$ and $n'=n\times2$ so as to obtain the quotient Q and the remainder R of the multiplication-division $(M_1\times M_2)\div n$, the least significant bit δ_0' of M_2' is always 0, so that it is not a difficult condition to cause Eq. (F13) to hold when $\omega=1$. However, since $R=[R_2\div2]$ and $Q=Q_2$ hold for the quotient Q_2 and the remainder R_2 of $(M_1\times M_2')\div n'$, the least significant bit of R_2 is unnecessary for R.

The addition of Eq. (F18) is applied to the case of using a carry save type adder, but this addition can also be performed using a carry propagation adder, with $\alpha_j=0$ and $A=1$. In the following, a constant may sometimes be called a parameter.

$$1 \leqq \frac{n}{2^{m+k}} \leqq 2^k \quad \text{(F1)}$$

$$0 \leqq \alpha_j \leqq A \quad \text{(F2)}$$
$$\lambda + \log_2 A \leqq K \quad \text{(F3)}$$
$$m = 0, 1, 2, \ldots \quad \text{(F4)}$$
$$\lambda = 1, 2, \ldots \quad \text{(F5)}$$
$$\omega = 0 \text{ or } 1 \quad \text{(F6)}$$

$$-2^k + 1 + \omega + I_1 \leqq S \leqq 2^{k+1} - A - 1 - 2\omega - (I_2 + 1) \quad \text{(F7)}$$
where $I_1 = [2^\lambda \cdot t_1 + 2] \geqq 1$
and $I_2 = [2^{\lambda+1} + 2^\lambda \cdot t_2] \geqq 0$

$$2^{m+1} - nt_2 + I_1 \cdot 2^m \leqq S \cdot 2^m \leqq nt_1 - \omega 2^m - (I_2 + 1)2^m \quad \text{(F8)}$$

$$0 \leqq M_1 < n \quad \text{(F9)}$$
$$M_{2j}' = M_{2j} - \omega\delta_0 \cdot 2^\lambda + \omega\delta_{(j-1)\lambda} \quad \text{(F10)}$$

where $M_2 = \sum_{j=1}^{l} M_{2j} \cdot 2^{(j-1)\lambda}$ (F11)

-continued

$$M_{2j} = \sum_{\mu=0}^{\lambda-1} \delta_{(j-1)\lambda+\mu} \cdot 2^{\mu},\ M_2 = \sum_{\mu=0}^{l\lambda-1} \delta_{\mu} \cdot 2^{\mu} \quad (F12)$$
$$\delta_{(j-1)\lambda+\mu} = 0 \text{ or } 1$$

$$\delta_0 = 0 \text{ but when } \omega = 1 \quad (F13)$$

$$R_{l+1} = 0 \quad (F14)$$

$$-A \cdot n \cdot 2^{-k} - t_1 \cdot n + \omega\delta_{\lambda} \cdot M_1 \leqq \quad (F15)$$
$$R_{j+1} \leqq n + t_2 \cdot n + \omega\delta_{\lambda} \cdot M_1$$

$$I_j = [\{[(2^{\lambda}R_{j+1})2^{-m}] + [(M_1 \cdot M'_{2,j})2^{-m}] + S + \alpha_j - \quad (F16)$$
$$\omega\delta_{(j-1)\lambda} \cdot [M_1 \cdot 2^{-m}]\} + [n \cdot 2^{-m}]]$$

$$-I_1 \leqq I_j \leqq I_2 \quad (F17)$$

$$R_j = 2^{\lambda} \cdot R_{j+1} + M_1 \cdot M'_{2,j} - I_j \cdot n - \quad (F18)$$
$$(S + A + (I_2 + 1)) \cdot 2^m + \omega \cdot \delta_{(j-1)\lambda} \cdot$$
$$[M_1 \cdot 2^{-m}] \cdot 2^m \leqq R_j$$

$$< n + (2 - S + I_1)2^m + \omega \cdot \delta_{(j-1)\lambda} \cdot [M_1 \cdot 2^{-m}] \cdot 2^m \quad (F19)$$

$$R = R_1 + \delta \cdot n \quad (F20)$$

$$Q = \sum_{j=1}^{l} I_j \cdot 2^{(j-1)\lambda} - \delta \quad (F21)$$

$$\delta = \begin{bmatrix} 2 \text{ for } R_1 < -n \\ 1 \text{ for } -n \leqq R_1 < 0 \\ 0 \text{ for } 0 \leqq R_1 < n \\ -1 \text{ for } n \leqq R_1 \leqq 2n \\ -2 \text{ for } 2n \leqq R_1 < 3n \end{bmatrix} \quad (F22)$$

Corollary of Theorem

Corollaries of the theorem will be given below on the assumption that the range of application of the theorem is extended. Here, a combination of corollaries 2 and 3 is impossible but a desired combination of other corollaries is possible; for example, corollaries 1, 2, 4 and 5 can be applied at the same time. Next, abridged notations X and X' and a random number β_j, which irregularly changes its value with variations in j, are defined by the following equations:

$$X = [(2^{\lambda} \cdot R_{j+1})2^{-m}] + [(M_1 \cdot M'_{2,j})2^{-m}] + S + \quad (F23)$$
$$\alpha_j - \omega \cdot \delta_{(j-1)\lambda} \cdot [M_1 \cdot 2^{-m}]$$

$$X' = [(2^{\lambda} \cdot R_{j+1,1})2^{-m}] + [(2^{\lambda} \cdot R_{j+1,0})2^{-m}] + \quad (F24)$$
$$\sum_{\mu=0}^{\phi-1} [(Z_{j,\mu})2^{-m}] + S' + \alpha_j$$

where

$$R_{j+1} = R_{j+1,1} + R_{j+1,0} \quad (F25)$$

$$M_1 \cdot M'_{2,j} = \sum_{\mu=0}^{\phi-1} Z_{j,\mu} \quad (F26)$$

$$S' = S + \beta_j \quad (F27)$$
$$0 \leqq \beta_j \leqq \phi \quad (F28)$$

where ψ and β_j are integers.

Corollary 1

I_j can be obtained with Eq. (F29) instead of Eq. (F16). But when Eq. (F29) is used, S in the theorem is replaced with S'. Incidentally, the use of a carry propagation adder for the addition of Eq. (F18) means that the application of the corollary 1 is meaningless.

$$I_j = [X + [n \cdot 2^{-m}]] \quad (F16)$$

$$I_j = [X' + [n \cdot 2^{-m}]] \quad (F29)$$

Corollary 2

Eqs. (F30) and (F31), but Eq. (F16) can be obtained from Eqs. (F32) to (F34) on the premise that w is an integer. In this case, however, Eqs. (F8), (F18), (F19) and (F21) become (F8)", (F18)", (F19)" and (F21)", respectively.

$$I_1 \leqq 2^w \quad (F30)$$
$$(I_2 + 1) \leqq 2^w \quad (F31)$$

$$I'_j = I_j + \delta^*_j = \begin{bmatrix} [X \cdot v \cdot 2^{-(k+w+1)}] + 1 \text{ for } X \geqq 0 \\ [X \cdot v \cdot 2^{-(k+w+1)}] \quad \text{for } X < 0 \end{bmatrix} \quad (F32)$$

$$v = [2^{(k+w+1)}] + [n \cdot 2^{-m}]] \quad (F33)$$
$$\delta^*_j = 0 \text{ or } 1 \quad (F34)$$

$$-\{S + A + (I_2 + 1)\}2^m + \omega \cdot \delta_{(j-1)\lambda}[M_1 \cdot 2^{-m}] \cdot \quad (F19)''$$
$$2^m - n \leqq R_j < n + (2 - S + I_1) \cdot 2^m + \omega \cdot$$
$$\delta_{(j-1)\lambda}[M_1 \cdot 2^{-m}] \cdot 2^m$$

$$2^{m+1} - n \cdot t_2 + I_1 \cdot 2^m \leqq S \cdot 2^m \leqq t_1 \cdot n - \omega \cdot 2^m - \quad (F8)''$$
$$(I_2 + 1) \cdot 2^m - n$$

$$R_j = 2^{\lambda} \cdot R_{j+1} + M_1 \cdot M'_{2,j} - (I_j + \delta^*_j) \cdot n \quad (F18)''$$

$$Q = \sum_{j=1}^{l} (I_j + \delta^*_j) \cdot 2^{(j-1)\lambda} - \delta \quad (F21)''$$

$$\delta = \begin{bmatrix} 3 \text{ for } R_1 < -2n \\ 2 \text{ for } -2n \leqq R_1 < -n \\ 1 \text{ for } -n \leqq R_1 < 0 \\ 0 \text{ for } 0 \leqq R_1 < n \\ -1 \text{ for } n \leqq R_1 < 2n \\ -2 \text{ for } 2n \leqq R_1 < 3n \end{bmatrix} \quad (F22)''$$

That is, a unit of I'_j is used instead of I_j.

Corollary 3

Instead of Eq. (F16) for calculating I_j, I_j can be obtained from Eqs. (F35) and (F36). In this case, however, let Eqs. (F7), (F8) and (F19) be Eq. (F7)', (F8)' and (F19)', respectively.

$$X + [(-I_j)n \cdot 2^{-m}] \geqq 0 \quad (F35)$$

$$X + [(-I_j - 1)n \cdot 2^{-m}] < 0 \quad (F36)$$

$$-2^k + 1 + \omega \leqq S \leqq 2^{k+1} - A - 1 - 2\omega \quad (F7)'$$

$$2^{m+1} - n \cdot t_2 \leqq S \cdot 2^m \leqq t_1 \cdot n - \omega \cdot 2^m \quad (F8)'$$

$$-(S + A) \cdot 2^m + \omega \cdot \delta_{(j-1)\lambda}[M_1 \cdot 2^{-m}]2^m \leqq R_j \quad (F19)'$$
$$< n + (2 - S)2^m + \omega \cdot \delta_{(j-1)\lambda}[M_1 \cdot 2^{-m}]2^m$$

Corollary 4

The quotient Q and the remainder R can be obtained, letting the lower and upper limit values of I_j expressed by Eq. (F17) be $-I_1+1$ if $A \cdot 2^{-k} + t_1 \geqq 0$ and $I_2 - 2$ if $t_2 \geqq 0$, respectively. In this case, I_1 in the theorem becomes $I_1 - 1$ and I_2 becomes $I_2 - 2$.

Corollary 5

By obtaining R_j from Eq. (F18) using $I_j'=I_j+I_{j0}'$ (where $I_{j0}'=\pm1, \pm2, \ldots$) for I_j obtained by Eq. (F16), the range of R_j is given by Eq. (37). If this range of R_j is included in the range of R_{j+1}, then the theorem holds. When the corollary 5 is combined with the corollary 3, the range of R_j is given by Eq. (F38) and when combined with corollary 4, the range is given by Eq. (F39). When this corollary is combined with both of the corollaries 3 and 4, the range of R_j is given by Eq. (F38). In the case where only one of the lower and upper limit values of I_j is used for the corollary 4, I_1-1 and $(I_2+2)-2$ in Eq. (F39) respectively become I_1 and (I_2+1) corresponding to the lower and upper limit values.

$$-(S+A+(I_2+1))\cdot 2^m+\omega\cdot\delta_{(j-1)\lambda}[M_1\cdot 2^{-m}]\cdot 2^m-I_{j0}\cdot n\leqq R_j<n+(2-S+I_1)\cdot 2^m+\omega\cdot\delta_{(j-1)\lambda}[M_1\cdot 2^{-m}]\cdot 2^m-I_{j0}\cdot n \quad (F37)$$

$$-(S+A)\cdot 2^m+\omega\cdot\delta_{(j-1)\lambda}[M_1\cdot 2^{-m}]\cdot 2^m-I_{j0}\cdot n\leqq R_j<n+(2-S)\cdot 2^m+\omega\cdot\delta_{(j-1)\lambda}[M_1\cdot 2^m]\cdot 2^m-I_{j0}\cdot n \quad (F38)$$

$$-(S+A+(I_2+1)-2)\cdot 2^m+\omega\cdot\delta_{(j-1)\lambda}[M_1\cdot 2^{-m}]\cdot 2^m-I_{j0}\cdot n\leqq R_j<n+(2-S+(I_1-1))2^m+\omega\cdot\delta_{(j-1)\lambda}[M_1\cdot 2^{-m}]\cdot 2^m-I_{j0}\cdot n \quad (F39)$$

SPECIFIC EXAMPLES OF EXPRESSIONS OF THEOREM AND COROLLARIES

The following will show by way of example that the principle of the multiplier-divider could be expressed in various forms by suitable definition of constants shown in the theorem and its corollaries. In the following, those expressions are omitted which would inevitably result from definition of the constants. As regards those equations which would become easy to understand by changing their numerical expressions, they are represented with their expression changed.

EXAMPLE 1

This is an example in which the constants K, A, λ, ω, S, t_1 and t_2, excepting m are K=7, A=1, λ=4, ω=0, S=26, $t_1=185/128$ and $t_2=0$. The corollaries used are the corollary 1, the corollary 2, where w=5, and the corollary 4.

$$1\leqq\frac{n}{2^{m+7}}<2$$

$$0\leqq\alpha_j\leqq 1$$
$$m=0, 1, 2, \ldots$$
$$0\leqq M_1<n$$

$$M_2=\sum_{j=1}^{1}M_{2j}\cdot 2^{4(j-1)}$$

where

$$M_{2j}=\sum_{\mu=0}^{3}\delta_{4(j-1)+\mu}\cdot 2,\ \delta_{4(j-1)+\mu}=0 \text{ or } 1$$

$$R_{j+1}=0-\frac{58}{128}n-n\leqq R_{j+1}<n$$

-continued

$$I_j'=I_j+\delta_j^*=\begin{cases}[X_1'\cdot v\cdot 2^{-13}]+1, & \text{for } X_1'\geqq 0\\ [X_1'\cdot v\cdot 2^{-13}], & \text{for } X_1'<0\end{cases}$$

$\delta_j^*=0$ or 1

where

$$X_1'=[(2^4\cdot R_{j+1,1})2^{-m}]+[(2^4\cdot R_{j+1,0})2^{-m}]+\sum_{\mu=0}^{3}[(M_1\cdot\delta_{4(j-1)+\mu}\cdot 2^\mu)2^{-m}]+26+\alpha_j+\beta_j$$

$0\leqq\beta_j\leqq 4$

$$v=\left[\frac{2^{13}}{[n\cdot 2^{-m}]}\right]-24\leqq I_j\leqq 30$$

$$R_j=2^4\cdot R_{j+1}+M_1\cdot M_{2j}-I_j'\cdot n-n-\frac{58n}{128}\leqq R_j<n$$

$$R=R_1+\delta\cdot n$$

$$Q=\sum_{j=1}^{1}I_j'\cdot 2^{4(j-1)}-\delta$$

$$\delta=\begin{cases}2 \text{ for } R_1<-n\\ 1 \text{ for } -n\leqq R_1<0\\ 0 \text{ for } R_1\geqq 0\end{cases}$$

EXAMPLE 2

This is an example in which I_1 and I_2 are determined using the corollary 4 and then the corollary 3 is applied.

(A) Precondition

The constants K, λ, ω, t_1, t_2 and S, excepting m and A, are determined by the following equations. ξ is a newly defined variable.

$$K\geqq 2$$

$$X=1$$

$$\omega=0$$

$$t_1=2^{-k+1}$$

$$t_2=0$$

$$2^k\geqq A+S$$

$$S=2$$

(B) Calculations of the Constants

By obtaining I_1 and I_2 using the corollary 4, $I_1=1$ and $I_2=2$ are obtained, and $-1\leqq I_j\leqq 2$ holds.

Next, defining a variable Q_j which equals I_j+1, it follows that $0\leqq Q_j\leqq 3$. Here, since the corollary 4 is applied, for example, when $Q_j\leqq 4$ is obtained, it is set that $Q_j=3$.

Setting

$$M_2=\sum_{\mu=0}^{q}\delta_\mu\cdot 2^\mu,$$

the relation $M_{2j}'=\delta_j$ holds on the conditions ω=0, λ=1. Therefore, the following equations holds by the corollary 3:

$$[(2\cdot R_{j+1})2^{-m}]+[(\delta_j M_1)2^{-m}]+2+\alpha_j$$

$$+[(1-Q_1)n\cdot 2^{-m}]\geq 0$$

$$[(2\cdot R_{j+1})2^{-m}]+[(\delta_j M_1)2^{-m}]+2+\alpha_j +[(-Q_j)n\cdot 2^{-m}]<0$$

Obtaining the range of R_j from Eq. (F19), the following equation is obtained using $S+A\leqq 2\xi$:

$$-n \leqq -\frac{n}{2^{k-\xi}} \leqq R_j < n$$

Eq. (F18) becomes as follows:

$$R_j=2R_{j+1}+\delta_j M_1+(1-I_j)n$$

Taking into account that j=l, l−1, . . . 1, 0, Eq. (F20) becomes as follows:

$$R=R_0+\delta_{3j}n$$

where

$$\delta_{3j}=\begin{cases}1 \text{ for } R_0<0\\ 0 \text{ for } R_0\geq 0\end{cases}$$

It will easily be understood that in the case of $\delta_{3j}=1$ the expected value of $\delta_{3j}=1$ becomes $2^{-k+\xi-1}$ on the assumption that R_0 is uniformly distributed in the section of

$$-\frac{n}{2^{k-\xi}} \leqq R_0 < n.$$

(C) Summary of the Constants and Equations

The following equation (H6) holds for R_j (where j=q, q−1, . . . 1, 0) defined by the following equations (H1) to (H5) shown as equations summarizing the above, and the quotient Q and the remainder R of $(M_1\times M_2)\div n$ are given by the following equation (H7), where when $\delta_j=1$ and $Q_j=3$ holds at the same time, it is regarded that Eq. (H2) and (H3) hold.

$$\left.\begin{array}{l} R_{l+1}=0\\ 0\leqq M_1<n\\ 1\leqq \dfrac{n}{2^{m+k}}<2\\ M_2=\sum_{j=0}^{q}\delta_j\cdot 2^j, \text{ where } \delta_j=0 \text{ or } 1\\ K\geq 2\\ m\geq 0\\ 2\geq 2+\alpha_j \end{array}\right\} \quad (H1)$$

$$[(2R_{j+1})\cdot 2^{-m}]+[(\delta_j\cdot M_1)\cdot 2^{-m}]+[(1-Q_j)\cdot n\cdot 2^{-m}]+2+\alpha_j\geq 0 \quad (H2)$$

$$[(2R_{j+1})\cdot 2^{-m}]+[(\delta_j\cdot M_1)\cdot 2^{-m}]+[(-Q_j)\cdot n\cdot 2^{-m}]+2+\alpha_j<0 \quad (H3)$$

$$0\leqq Q_j\leqq 3 \quad (H4)$$

$$R_j=2R_{j+1}+\delta_j\cdot M_1+(1-Q_j)\cdot n \quad (H5)$$

$$-\frac{n}{2^{k-\xi}} \leqq R_j < n \quad (H6)$$

$$R=R_0+\delta_{3j}\cdot n$$

$$Q=\sum_{j=0}^{q}(Q_j-1)2^j-\delta_{3j}$$

where

-continued

$$\delta_{3j}=\begin{cases}1 \text{ for } R_0<0\\ 0 \text{ for } R_0\geq 0\end{cases} \quad (H7)$$

Mean value is $2^{-k+\xi-1}$ when $\delta=1$.

EXAMPLE 3

This is an example in which the corollaries 1, 2 and 4 are employed and, when j=1, the corollary 5 is further used, and in which the constants K, A, λ, ω, S, t_1, and t_2, excepting m, are set as follows: K=11, A=1, λ=8, ω=1, S=405, $t_1=1+1173/2048$ and $t_2=1$. Moreover, the value of w in the corollary 2 is set as w=10.

$$1\leqq n\cdot 2^{-m-11}<2$$
$$\alpha_j=0, 1$$
$$m=0, 1, 2, \ldots$$
$$0\leqq M_1<n$$
$$M_{2j}=M_{2j}-2^8\cdot\delta_{8j}+\delta_{8(j-1)}$$

where

$$M_2=\sum_{j=1}^{l}M_{2j}\cdot 2^{8(j-1)}$$

$$M_{2j}=\sum_{\mu=0}^{7}\delta_{8(j-1)+\mu}\cdot 2^{\mu},\ \delta_{8(j-1)+\mu}=0 \text{ or } 1$$

$$\delta_0=0$$
$$R_{l+1}=0$$

$$-\frac{1174}{2048}\cdot n-n+\delta_{8j}\cdot M_1\leqq R_{j+1}<2n+\delta_{8j}\cdot M_1$$

$$X_2'=[(2^8\cdot R_{j+1,1})2^{-m}]+[(2^8\cdot R_{j+1,0})2^{-m}]+\sum_{\mu=0}^{7}[(Z_{j,\mu})2^{-m}]+S'+\alpha_j$$

where

$$R_{j+1,1}+R_{j+1,0}=R_{j+1}$$

$$\sum_{\mu=0}^{7}Z_{j,\mu}=M_1\cdot\{-\delta_{8j}2^8+\delta_{7j+7}\cdot 2^7+\ldots+\ldots+\delta_{7j+1}\cdot 2^L+\delta_{7j}\cdot 2^0\}$$

$$S'=405+\beta_j$$
$$0\leqq\beta_j\leqq 9$$

$$Ij=I_j+\delta'=\begin{cases}[X_2'\cdot\nu\cdot 2^{-22}]+1 \text{ for } X_2'\geq 0\\ [X_2'\cdot\nu\cdot 2^{-22}] \text{ for } X_2'<0\end{cases}$$

$$\delta'=0, 1$$
$$\nu=[2^{22}+[n\cdot 2^{-m}]]$$
$$-403\leqq I_j\leqq 766$$
$$R_j=2^8\cdot R_{j+1}+M_1\cdot M'_{2j}-Ij\cdot n$$

$$-\frac{1173}{2048}\cdot n+\delta_{8(j-1)}[M_1\cdot 2^{-m}]\cdot 2^m\leqq R_j<n+\delta_{8(j-1)}\times[M_1\cdot 2^{-m}]2^m$$

$$R=R_1+\delta\cdot n$$

$$Q=\sum_{j=0}^{l}Ij\cdot 2^{8(j-1)}-\delta$$

$$\delta=0, 1, 2, 3, 0\leqq R_1+\delta\cdot n<n$$

EXAMPLE 4

This is an example in which the values of I_1 and I_2 are defined using the corollary 4 and then the corollaries 1 and 4 are applied.

(A) Precondition

At first, the constants are defined by the following equations, in which X_j'', Q_j'' and $Z_{j,\mu}$ are constants to be newly defined.

$$K=\lambda+2$$

L=the number of significant digits,

i.e.

$$2^{L-1}\leqq n<2^L$$

$$I_1=2-2^{-K}$$

$$I_2=0$$

A two-output carry save adder is used.

That is, $A=1$, $\alpha_j=0, 1$

$$S \geqq I_1 + 2$$
$$X_j'' = X''$$

$$Q_j'' I_j = I_j + \delta_j^{\alpha}$$

$$\phi = \begin{cases} \lambda \text{ for } \omega = 0 \\ \lambda/2 \text{ for } \omega = 1 \end{cases}$$

$$Z_{j,\mu} = M_1 \cdot \delta_{(j-1)\lambda+\mu} \cdot 2^{\mu} \text{ for } \omega = 0$$

$$Z_{j,\mu} = -M_1 \cdot \delta_{(j-1)\lambda+2\mu+2} \cdot 2^{2\mu+2} + M_1 \cdot \delta_{(j-1)\lambda+2\mu+1} \cdot 2^{2\mu+1} + 2M_1 \cdot \delta_{(j-1)\lambda+2\mu} \cdot 2^{2\mu} \text{ for } \omega = 1$$

$$u = K + W + 1$$

(B) Calculation of the Constants

I_1 and I_2 obtained using the corollary 4 are as follows:

$$I_1=2^{\lambda+1},\ I_2=2^{\lambda+1}-2$$

From Eq. (F1), $L=m+K+1 \therefore m=L-\lambda-3$

The range of S is obtained from Eqs. (F7) and (F8). But the range of S is made smaller than that obtained by calculation and ω is eliminated.

$$2^{\lambda+1}+2 \leqq S \leqq 2^{\lambda+2}$$

From Eqs. (F15) and (F19) the ranges of R_{j+1} and R_j are obtained. The following is simplified representation of the ranges of R_{j+1} and R_j using the condition

$$M_1=[M_1 \cdot 2^{-m}] \cdot 2^m + 2^m \cdot \epsilon,$$

where

$$0 \geqq \epsilon < 1,\ [M_1 \cdot 2^{-m}] \cdot 2^m \leqq M_1.$$

$$-2n+\omega-\delta_{j\lambda} \cdot M_1 \leqq R_{j+1} < n+\omega-\delta_{j\lambda} \cdot M_1$$

$$-n+\omega-\delta_{(j-1)\lambda} \cdot M_1 < R_j < n+\omega-\delta_{(j-1)\lambda} \cdot M_1$$

From the corollary 2,

$$\left.\begin{array}{l} I_1 \leqq 2^w \\ I_2 + 1 \leqq 2^w \end{array}\right\} \begin{array}{l} \therefore w \geqq \lambda + 1 \\ \therefore u \geqq 2\lambda + 4 \end{array}$$

$$Q_j'' = \begin{cases} [X_j'' \cdot v \cdot 2^{-u}] + 1 \text{ for } X_j'' \geqq 0 \\ [X_j'' \cdot v \cdot 2^{-u}] \text{ for } X_j'' < 0 \end{cases}$$

-continued

$$\text{where } v = \left[\frac{2^u}{[n \cdot 2^{-m}]}\right]$$

From Eq. (F17), $-2^{\lambda+1} \leqq I_j \leqq 2^{\lambda+1}-2$.

Since $Q_j''=I_j''=I_j+\delta_j^{\alpha}$, where $\delta_j^{\alpha}=0$ or 1, the range of Q_j'' is defined by $-2^{\lambda+1} \leqq Q_j'' \leqq 2^{\lambda+1}$. However, when Q_j'' obtained from Eq. (F32) is $Q_j''=-2^{\lambda+1}$, then Q_j'' may be set as $Q_j''=-2^{\lambda+1}+1$ and when $Q_j''=2^{\lambda+1}$, then Q_j'' may be set as $Q_j''=2^{\lambda+1}-1$. As a result, the range of Q_j'' may be defined as follows:

$$-2^{\lambda+1}+1 \leqq Q_j'' \leqq 2^{\lambda+1}-1$$

But when Q_j'' obtained from Eq. (F32) is $Q_j''=-2^{\lambda+1}$, set $Q_j''=-2^{\lambda+1}+1$, and when $Q_j''=2^{\lambda+1}$, set $Q_j''=2^{\lambda+1}-1$.

(C) Summary of Calculation Method of Parameters

At first L, λ, l, and ω are determined and then a set of integers n, S and u is obtained.

L=the number of significant digits (the number of bits)

λ=the length of division of M_2

l=ceil {L+λ}

ω=0 or 1

$m=L-\lambda-3$

$2^{\lambda+1}+2 \leqq S \leqq 2^{\lambda+2}$

$u \geqq 2\lambda+4$

where when $\omega=1$, λ is an even number. ceil {x} indicates a minimum integer greater than x; for example, ceil {1.5}=2.

(D) Execution of Calculation

(a) Preparation

At first, n is input to obtain v. $v=[2^u \div [n \cdot 2^{-m}]]$

Next, M_1 and M_2 are input.

(b) Repeated Calculation

The calculation method is shown below in the form of a program flowchart.

Step 0:

$$j \leftarrow l,\ R_{l+1,l} \leftarrow 0,\ R_{l+1,i} \leftarrow 0$$

Step 1:

$$X_j'' \leftarrow \sum_{i=0}^{\lambda-1} [(2^{\lambda} \cdot R_{j+1,i}) \cdot 2^{-m}] + \sum_{\mu=0}^{\phi-1} [(z_{j,\mu}) \cdot 2^{-m}] + S + \alpha_j + \beta_j$$

where $-2^u < X_j'' < 2^u$

Step 2:

$$Q''_j \leftarrow \begin{cases} [X''_j \cdot v \cdot 2^{-u}] + 1 \text{ for } X''_j \geqq 0 \\ [X''_j \cdot v \cdot 2^{-u}] \text{ for } X''_j < 0 \end{cases}$$

where when $Q_j''=-2^{\lambda+1}$, set $Q_j''=-2^{\lambda+1}+1$, when $Q_j''=2^{\lambda+1}$, set $Q_j''=2^{\lambda+1}-1$.

Step 3:

$$\sum_{i=0}^{1} R_{j,i} \leftarrow \sum_{i=0}^{1} 2^{\lambda} \cdot R_{j+1,i} + \sum_{\mu=0}^{\phi=0} Z_{j,\mu} - Q''_j \cdot n$$

Step 4:
When j=1, go to step 5.
Then j←j=1, go back to step 1.
Step 5: Repeated Calculation ends.
(c) Repeated Calculation
Step 6:

$$R_1 \leftarrow \sum_{i=0}^{1} R_{1,i}$$

If $R_1 \geqq 0$, then go to step 8.
Step 7:

$$\sum_{i=0}^{1} R_{1,i} \leftarrow \sum_{i=0}^{1} R_{1,i} + n$$

Go back to step 6.
Step 8: When $R_1 \geqq \omega \cdot n$, $R_1 \leftarrow R_1 - n$ $R \leftarrow R_1$. Halt.

EXAMPLE 5

This is an example in which $K=\lambda+3$ is set in place of $K=\lambda+2$ in Example 4. The other conditions are the same as those in Example 4. Only differences between the two example are given below.

$$K=\lambda+3$$

$$m=L-\lambda-4$$

$$\mu \geqq 2\lambda+5$$

The aforementioned embodiment of the present invention is described in connection with the case where L=512, λ=4, l=128 and ω=0 and m=504, S=38 and u=13 are adopted.

Furthermore, it will easily be seen that, by setting $K=\lambda+i'$, $i'=4, 5, \ldots$, such various expressions as described above in respect of Examples 4 and 5 can be obtained.

VERIFICATION OF NUMERICAL EXPRESSIONS OF THE PRINCIPLE OF THE MULTIPLIER DIVIDER

Verificaton of Theorem Preparation of Verification

The following are definitions of abridged numerical expressions:

$$R_{-m}=[(2^{\lambda} \cdot R_{j+1})2^{-m}] \quad \text{(F67)}$$

$$M_{-m}=[(M_1 \cdot M_{2,j})2^{-m}] \quad \text{(F68)}$$

$$\omega_{-m}=[(-\omega \cdot \delta_{\mu} 2^{\lambda} \cdot M_1)2^{-m}] \quad \text{(F69)}$$

$$\omega_{-m-1}=[(\omega-\delta_{\mu-1})\lambda \cdot M_1)2^{-m}] = \omega-\delta_{\mu-1})\lambda[M_1 \cdot 2^{-m}] \quad \text{(F70)}$$

From Eq. (F15),

$$-A \cdot n \cdot 2^{-h+\lambda} - 2^{\lambda} \cdot i_1 \cdot n + \omega \cdot \delta_{\mu} 2^{\lambda} M_1 \leqq 2^{\lambda} R_{j+1} < 2^{\lambda} \cdot n + 2^{\lambda} i_2 \cdot n + \omega \cdot \delta_{\mu} 2^{\lambda} M_1 \quad \text{(F71)}$$

Setting $h=K-\lambda-\log_2 A$, from Eq. (F3),

$$h \geqq 0 \quad \text{(F72)}$$

Omitting low-order m bits on either side of Eq. (F71),

$$[(-n \cdot 2^{-h} - 2^{\lambda} \cdot i_1 \cdot n + \omega - \delta_{\mu} 2^{\lambda} M_1)2^{-m}] \leqq R_{-m} \quad \text{(F73)}$$

$$R_{-m} < [(2^{\lambda} \cdot n + 2^{\lambda} i_2 \cdot n + \omega - \delta_{\mu} 2^{\lambda} M_1)2^{-m}] \quad \text{(F74)}$$

On the other hand, the following equation holds for the real number x_j and an integer ϕ where δ_{ϕ} is an integer.

$$\sum_{j=1}^{\phi} [x_j] = \left[\sum_{i=1}^{\phi} x_i\right] - \gamma_{\phi} \quad \text{(F75)}$$

$$0 \leqq \gamma_{\phi} \leqq \phi - 1 \quad \text{(F76)}$$

When $\omega \neq 0$, applying Eq. (F75) to $M_1 \times$ Eq. (F10),

$$M_{-m}=[(M_1 \cdot M_{2,j})2^{-m}]+\omega_{-m}+\omega_{-m-1}+\gamma_3 \cdot \omega \quad \text{(F77)}$$

$$\gamma_3=0, 1 \text{ or } 2 \quad \text{(F78)}$$

From Eq. (F9)$\times M_{2,j}$

$$0 \leqq [(M_1 \cdot M_{2,j})2^{-m}] < [(n \cdot M_{2,j})2^{-m}] \quad \text{(F79)}$$

From Eqs. (F77) and (F79),

$$\omega_{-m}+\omega_{-m-1}+\gamma_3 \cdot \omega \leqq M_{-m} \quad \text{(F80)}$$

$$M_{-m} < \omega_{-m}+\omega_{-m-1}+\gamma_3 \cdot \omega+[(n \cdot M_{2,j})2^{-m}] \quad \text{(F81)}$$

Next, substituting Eqs. (F67), (F68) and (F70) into Eq. (F16)

$$l_j = \left[\frac{(R_{-m} + M_{-m} + S + \alpha_j - \omega_{-m-1})}{[n \cdot 2^{-m}]}\right] \therefore \quad \text{(F82)}$$

$$\frac{(R_{-m} + M_{-m} + S + \alpha_j - \omega_{-m-1})}{[n \cdot 2^{-m}]} - 1 < l_j \leqq \frac{(R_{-m} + M_{-m} + S + \alpha_j - \omega_{-m-1})}{[n \cdot 2^{-m}]} \therefore$$

$$R_{-m} + M_{-m} + S + \alpha_j - \omega_{-m-1} + (-l_j)[n \cdot 2^{-m}] \geqq 0 \quad \text{(F83)}$$

$$R_{-m} + M_{-m} + S + \alpha_j - \omega_{-m-1} + (-l_j - 1) \cdot [n \cdot 2^{-m}] < 0$$

On the other hand, the following equation holds for the integer I and the real number x:

$$[I+x]=I+[x] \quad \text{(F84)}$$

Accordingly, removing the Gaussian symbols from the both sides of Eqs. (F82) and (F83) and omitting the decimal point, Eqs. (F82) and (F83) become the following equations because R_{-m}, S, α_j and ω_{-m-1} are respectively integers.

$$R_{-m}+M_{-m}+S+\alpha_j-\omega_{-m-1}+[(-l_j)[n \cdot 2^{-m}]] \geqq 0 \quad \text{(F82)'}$$

$$R_{-m}+M_{-m}+S+\alpha_j-\omega_{-m-1}+[(-l_j-1)[n \cdot 2^{-m}]] < 0 \quad \text{(F83)'}$$

However, the following equation holds for the integer I and the real number $x>0$, with P an integer.

$$[(-I)-[x]]=[(-I)-x]+P \quad \text{(F85)}$$

where

$$\left.\begin{array}{l} 0 \leqq P \leqq I \text{ for } I \geqq 0 \\ I \leqq P \leqq 0 \text{ for } I \leqq 0 \end{array}\right\} \quad \text{(F86)}$$

The following is assumed letting I_1' and I_2' be integers as condition of I_j.

$$-I_1' \leqq I_j \leqq I_2' \quad \text{(F87)}$$

$$\left.\begin{array}{l} -I_1' \leqq -1 \\ I_2' \geqq 0 \end{array}\right\} \quad \text{(F88)}$$

(A) When $I_j<0$:

Eqs. (F85) to (F88) are applied to Eqs. (F82)' and (F83)'.

$$R_{-m}+M_{-m}+S+\alpha_j-\omega_{-m-1}+[-(-I_j)n\cdot 2^{-m}]+P_1 \geqq 0 \quad \text{(F89)}$$

$$R_{-m}+M_{-m}+S+\alpha_j-\omega_{-m-1}+[(-I_j-1)n\cdot 2^{-m}]-P_1<0 \quad \text{(F90)}$$

$$-I_1' \leqq P_1 \leqq 0 \quad \text{(F91)}$$

(B) When $I_j \geqq 0$:

The following equations are obtained in the same manner as described above.

$$R_{-m}+M_{-m}+S+\alpha_j-\omega_{-m-1}+[-(-I_j)n\cdot 2^{-m}]+P_2 \geqq 0 \quad \text{(F92)}$$

$$R_{-m}+M_{-m}+S+\alpha_j-\omega_{-m-1}+[-(-I_j-1)n\cdot 2^{-m}]+P_2<0 \quad \text{(F93)}$$

$$0 \leqq P_2 \leqq I_2'+1 \quad \text{(F94)}$$

Lower Limit Value of I_j (left side of Eq. (F17))

Using U for the left side of Eq. (F89) and substituting with $I_j=-2^{\lambda}t_1-2$,

$$R_{-m}+M_{-m}+S+\alpha_j-\omega_{-m-1}+[(2^{\lambda}t_1+2)n\cdot 2^{-m}]-+P_1=U \quad \text{(F95)}$$

From Eq. (F73)+Eq. (F88)+Eq. (F95),

$$[(-n\cdot 2^{-h}-2^{\lambda}\cdot t_1\cdot n+\omega\cdot\delta_{\beta\cdot}2^{\lambda}M_1)2^{-m}]+\omega_{-m}+\omega_{-m-1}+\gamma_3\cdot\omega+S+\alpha_j-\omega_{-m-1}+P_1+[(2^{\lambda}t_1+2)n\cdot 2^{-m}]\leqq U$$

Applying Eq. (F75) to the above equations in the cases of $\omega=0$ and $\omega\neq 0$ separately,

$$[(1-2^{-h})n\cdot 2^{-m}+n\cdot 2^{-m}]-\gamma_2+\gamma_3\cdot\omega+S+\alpha_j+P_1\leqq U \quad \text{(F96)}$$

$$\gamma_2=0, \ldots 1+\omega \quad \text{(F97)}$$

$$\therefore[(1-2^{-h})n\cdot 2^{-m}]+[n\cdot 2^{-m}]+\epsilon_1-\gamma_2+\gamma_3\omega+S+\alpha_j+P_1\geqq U \quad \text{(F98)}$$

where,

$$\epsilon_1=0 \text{ or } 1 \quad \text{(F99)}$$

From Eq. (F72), $[(1-2^{-h})n\cdot 2^{-m}]\geqq 0$, and from Eq. (F1) $2^K\leqq[n\cdot 2^{-m}]$. Therefore, when the left side of Eq. (F98) becomes minimum when $\epsilon_1=0$, $\gamma_2=1+\omega$, $\gamma_3=0$, $\alpha_j=0$, $P_1=-I_1'$.

$$\therefore 2^K-(1+\omega)+S-I_1'\leqq U \quad \text{(F100)}$$

Accordingly, if I_1' is selected such that $I_1'=[2^{\lambda}t_1+2]\geqq 1$, the condition of Eq. (F88) is satisfied by Eq. (F7) and $I_1'=I_1$, resulting in the following equation holding:

$$0\leqq U \quad \text{(F101)}$$

Further, when $I_j<-2^{\lambda}t_1-2$, the Eq. (F89) holds but Eq. (F90) does not hold.

$$\therefore -I_1\leqq I_j \quad \text{(F102)}$$

Upper Limit Value of I_j (right side of Eq. (F17))

Using V for the left side $I_j=2^{\lambda+1}+2^{\lambda}\cdot t_2$ in Eq. (F93), it follows from $2^{\lambda+1}=2^{\lambda}+2^{\lambda}$ that

$$V=R_{-m}+M_{-m}+S+\alpha_j-\omega_{-m-1}+[(-2^{\lambda}n-2^{\lambda}n-2^{\lambda}t_2-n)2^{-m}]+P_2 \quad \text{(F103)}$$

Eq. (F74)+Eq. (F81)+Eq. (F103)

$$V<[2^{\lambda}n+2^{\lambda}t_2\cdot n+\omega\cdot\delta_{\beta}2^{\lambda}M_1]2^{-m}]+ \quad \text{(F104)}$$

$$\omega_{-m}+\omega_{-m-1}+\gamma_3\cdot\omega[(n\cdot M_{2j})2^{-m}]+$$

$$S+\alpha_j-\omega_{-m-1}+P_2+[(-2^{\lambda}n-2^{\lambda}n-2^{\lambda}\cdot t_2-n)2^{-m}]$$

where $\gamma_3=0$, 1 or 2 (F105)

Eq. (F75) of the formula is applied to Eq. (F105)

$$V<[M_{2j}-(2^{\lambda}-1))n\cdot 2^{-m}+(-2n)2^{-m}]- \quad \text{(F106)}$$

$$\gamma_4+\gamma_3\quad\omega+S+\alpha_j+P_2$$

$$\therefore V<[M_{2j}-\times(2^{\lambda}-1))n\cdot 2^{-m}]+$$

$$[(-2n)2^{-m}]+\epsilon_1-\gamma_4+\gamma_3\cdot\omega+S+\alpha_j+P_2$$

where

$$\epsilon_1=0 \text{ or } 1 \quad \text{(F107)}$$

$$\gamma_4=0, 1, 2 \text{ or } 3 \quad \text{(F108)}$$

$$\gamma_3=0, 1 \text{ or } 2 \quad \text{(F78)'}$$

However, $[(-2n)2^{-m}]\leqq -2^{k+1}$ holds from Eq. (F1) and $M_{2j}\leqq(2^{\lambda-1})$, that is, $[(M_{2j}-(2^{\lambda}-1))n\cdot 2^{-m}]\leqq 0$ holds from Eq. (F12). Since the right side of Eq. (F106) becomes maximum when $\epsilon_1=1$, $\gamma_4=0$, $\gamma_3=2$, $\alpha_j=A$, $P_2=I_2'+1$,

$$V<-2^{k+1}+1+2\omega+S+A+I_2'+1 \quad \text{(F109)}$$

Accordingly, if I_2' is selected such that $I_2'=[2^{\lambda+1}+2^{\lambda}\cdot t_2]\geqq 0$, then the condition of Eq. (F88) is satisfied by Eq. (F7) and, from Eq. (F7), the following equation holds with $I_2'=I_2$.

$$V<0 \quad \text{(F110)}$$

On the other hand, when $I_j\leqq 2^{\lambda+1}+2^{\lambda}\cdot t_2$, Eq. (F93) holds but Eq. (F92) does not.

$$\therefore I_j\leqq I_2 \quad \text{(F111)}$$

Range of R_j (Verification of Eq. (F19))

Eq. (F75) is applied to Eqs. (F89) and (F92) to substitute therein Eq. (F18).

$$[(R_j)2^{-m}]-\gamma_3+S+\alpha_j-\omega_{-m-1}+P\geq 0 \quad \text{(F112)}$$

where

$$\gamma_3=0, 1 \text{ or } 2 \quad \text{(F113)}$$

$$P=\begin{cases}P_1 \text{ for } I_j<0\\ P_2 \text{ for } I_j\geq 0\end{cases} \quad \text{(F114)}$$

$$\therefore \gamma_3-S-\alpha_j-P+\omega_{-m-1}\leq [(R_j)2^{-m}]$$

The left side of the above equation becomes minimum when $\gamma_3=0$, $\alpha_j=A$, $P=P_2=I_2+1$.

$$\therefore -(S+A+(I_2+1))+\omega_{-m-1}\leq [(R_j)2^{-m}] \quad \text{(F115)}$$

$$\therefore -(S+A+(I_2+1))\cdot 2^m+\omega\cdot\delta_{(j-1)\Lambda}[M_1\cdot 2^{-m}]\cdot 2^m\leq R_j$$

Next, Eq. (F75) to Eqs. (F90) and (F93) to substitute thereinto Eq. (F18).

$$[(R_j-n)2^{-m}]-\gamma_3+S+\alpha_j-\omega_{-m-1}+P<0 \quad \text{(F116)}$$

where

$$\gamma_3=0, 1 \text{ or } 2 \quad \text{(F117)}$$

$$P=\begin{cases}P_1 \text{ for } I_j<0\\ P_2 \text{ for } I_j\geq 0\end{cases} \quad \text{(F118)}$$

$$\therefore [(R_j-n)2^{-m}]<\gamma_3-S-\alpha_j-P+\omega_{-m-1}$$

The right side of the above equation becomes maximum when $\gamma_3=2$, $\alpha_j=0$, $P=P_1=-I_1$.

$$\therefore [(R_j-n)2^{-m}]<2-S+I_1+\omega_{-m-1} \quad \text{(F119)}$$

$$\therefore R_j-n<(2-S+I_1)\cdot 2^m+\omega_{-m-1}\cdot 2^m$$

$$\therefore R_j<n+(2-S+I_1)\cdot 2^m+\omega\cdot\delta_{(j-1)\Lambda}[M_1\cdot 2^{-m}]\cdot 2^m$$

By Eqs. (F115) and (F119) it has been verified that Eq. (F19) holds.

RELATIONSHIP BETWEEN RANGES OF R_{j+1} AND R_j ((F15) AND (F19))

Next, the fact that the range of R_j given by Eq. (F19) is smaller than the range of R_{j+1} given by Eq. (F15) is shown using j+1 for j and δ_Λ for $\delta_{(j-1)\Lambda}$ in Eq. (F19).

Set R_U=(upper limit value of R_{j+1}−upper limit value of R_j).

$$R_U=n+t_2\cdot n+\omega\cdot\delta_\Lambda M_1-n-(2-S+)I_1)\cdot 2^m- \quad \text{(F120)}$$
$$\omega\cdot\delta_\Lambda[M_1\cdot 2^{-m}]2^m$$

$$\therefore R_U=(S\cdot 2^m-2^{m+1}+n\cdot t_2-I_1\cdot 2^m)+$$
$$2^m(\omega\cdot\delta_\Lambda M_1\cdot 2^{-m}-[\omega\cdot\delta_\Lambda M_1\cdot 2^{-m}])$$

Therefore, from Eq. (F8)

$$R_U\geq 0 \quad \text{(F121)}$$

Set R_L=(lower limit value of R_j−lower limit value of R_{j+1}).

$$R_L=-(S+A+(I_2+1))\cdot 2^m+\omega\,\delta_\Lambda[M_1\cdot 2^{-m}]\cdot 2^m+$$
$$A\cdot n\cdot 2^{-k}+t_1\cdot n-\omega\cdot\delta_\Lambda M_1$$

$$\therefore R_L=(-S\cdot 2^m+t_1\cdot n-\omega\cdot 2^m-(I_2+1)\cdot 2^m)+$$
$$A\cdot 2^m(n\times 2^{-m-l}-1)+2^m[(\omega\cdot\delta_\Lambda\cdot M_1\cdot 2^{-m}]-$$
$$(\omega\cdot\delta_\Lambda\cdot M_1\cdot 2^{-m})+\omega\}$$

Therefore, from Eq. (F8)

$$R_L\geq 0 \quad \text{(F122)}$$

By Eqs. (F121) and (F122) it has been verified that the range of R_j is included in the range of R_{j+1}.

Accordingly, the recurrence formula shown by Eqs. (F14) to (F19) repeat in order j=l, l−1, . . . 2, 1, and R_j, R_{j-1}, . . . , R_2, R_1 and I_j, I_{j-1}, . . . I_2, I_1 can be obtained.

CALCULATION OF R AND Q (EQS. (F120) TO (F122))

The following equation is obtained by performing an operation

$$\sum_{j=1}^{l}\cdot 2^{(j-1)\lambda}$$

on both sides of Eq. (F20).

$$R_1=R_{l+1}\cdot 2^{l\lambda}+M_1\times\sum_{j=1}^{l}M_{2j}\cdot 2^{(j-1)\lambda}-\omega\cdot\delta_\Lambda M_1\cdot 2^{l\lambda}+$$
$$\omega\cdot\delta_0\cdot M_1-n\cdot\sum_{j=1}^{l}I_j\cdot 2^{(j-1)\lambda}$$

Accordingly, substituting $R_{l+1}=0$, $\delta_\Lambda=0$, $\delta_0=0$ and M_2 from Eqs. (F11) to (F14) to the above equation,

$$R_1=M_1\times M_2-n\times Q_1 \quad \text{(F123)}$$

where

$$Q_1=\sum_{j=1}^{l}I_j\cdot 2^{(j-1)\lambda} \quad \text{(F124)}$$

On the other hand, the following equation holds for the quotient Q and the remainder R of $(M_1\times M_2)\div n$.

$$R=M_1\times M_2-n\times Q \quad \text{(F125)}$$

From Eqs. (F123) and (F125) it follows that

$$R-R_1=-n\times(Q-Q_1) \quad \text{(F126)}$$

Since Q and Q_1 are integers, it is seen that the difference between R and R_1 is an integral multiple of n. And R_1 satisfies Eq. (F19), but I_1 and I_2 are substituted thereinto.

$$\therefore -\frac{(S+A+[2^{\lambda+1}+2^\lambda t_2+1])2^m+\omega\delta_0[M_1\cdot 2^{-m}]}{2^m}\leq R_1<n+\frac{(2-S+[2^\lambda t_1+2])\cdot 2^m+\omega\delta_0[M_1\cdot 2^{-m}]}{2^m}$$

However, since $0<\omega-\delta_0[M_1\cdot 2^{-m}]\cdot 2^m<n\omega$ holds from Eq. (F9), the following equation can be obtained.

$$-(S+A+[2^{\lambda+1}+2^{\lambda-t_2+1}])2^m\leqq R_1<2n+(2-S+[2^{\lambda-t_1+2}])\cdot 2^m+n\omega \quad (F127)$$

Then, the following equation is obtained from $2^m\times$Eq. (F7)

$$(-2^{k+1}+1+2\omega)\cdot 2^m\leqq -(S+A+[2^{\lambda+1}+2^{\lambda-t_2+1}]-)\cdot 2^m \quad (F128)$$

$$(2-S+[2^{\lambda-t_1+2}])2^m\leqq(2^k+1-\omega)2^m \quad (F129)$$

From Eqs. (F127) to (F129) it follows that

$$(-2^{k+1}+1+2\omega)\cdot 2^m\leqq R_1<n+(2^k+1-\omega)\cdot 2^m+n\omega \quad (F130)$$

Further, $-2n\leqq -2^{m+k+1}$, $2^{m+k}\leqq n$ hold from Eq. (F1) and, substituting them into the above equation,

$$-2n+(1+2\omega)\cdot 2^m\leqq R_1<2n+(1-\omega)\cdot 2^m+n\omega$$

From Eq. (F6), $\omega=0$ or 1 and, substituting it into the left side of the above equation,

$$-2n<R_1<2n+(1-\omega)\cdot 2^m+n\omega \quad (F131)$$

Since the difference between R and R_1 is an integral multiple of n, it is easily seen from Eqs. (F125) and (F126) that Eqs. (F20) to (F22) hold. Thus, it has been entirely verified that the theorem holds.

VERIFICATION OF COROLLARIES OF THEOREM

Verification of Corollary 1

From Eq. (F25)

$$[(2^{\lambda}\cdot R_{j+1})2^{-m}]=[(2^{\lambda}\cdot R_{j+1,1})2^{-m}]+[(2^{\lambda}R_{j+1,0})2^{-m}]+\beta_{j,R} \quad (F132)$$

where, $0\leqq\beta_{j,R}\leqq 1$

From Eq. (F26)

$$\left[\left(\sum_{\mu=0}^{\phi-1}Z_{j,\mu}\right)2^{-m}\right]=[(M_1\cdot M'_{2,j})2^{-m}] \quad (F133)$$

$$\left[\left(\sum_{\mu=0}^{\phi-1}Z_{j,\mu}\right)2^{-m}\right]=\sum_{\mu=0}^{\phi-1}[(Z_{j,\mu})2^{-m}]+\beta'_{j,z} \quad (F134)$$

where

$$0\leqq\beta_{j,z}\leqq\phi-1$$

From Eqs. (F132), (F133) and (F134)

$$[(2^{\lambda}\cdot R_{j+1})2^{-m}]+[(M_1\cdot M'_{2,j})\cdot 2^{-m}]=[(2^{\lambda}\cdot R_{j+1,1})2^{-m}]+[(2^{\lambda}\cdot R_{j+1,0})2^{-m}]+\sum_{\mu=0}^{\phi-1}[(Z_{j,\mu})2^{-m}]+\beta_j \quad (F136)$$

where $\beta_j=\beta_{j,R}+\beta'_{j,z}$

$$0\leqq\beta_j\leqq\phi \quad (F137)$$

Substituting Eqs. (F136) and (F137) into Eq. (F23), the following equation is obtained taking Eq. (F27) into account.

$$X=X' \quad (F138)$$

Accordingly, it is understood from Eqs. (F16) and (F138) that Eq. (F29) holds. Incidentally, when Eq. (F29) is used, S' is used, so that it is apparent that S' is used instead of S in the theorem.

Verification of Corollary 2

Substituting X in Eq. (F23) into Eqs. (F82) and (F83),

$$X+(-1_j)\cdot[n\cdot 2^{-m}]\geqq 0 \quad (F82)'$$

$$X+(-1_j-1)\cdot[n\cdot 2^{-m}]<0 \quad (F83)'$$

$$\therefore(1_j)\cdot[n\cdot 2^{-m}]\leqq X<(1_j+1)\cdot[n\cdot 2^{-m}] \quad (F139)$$

By substituting the minimum and the maximum value of 1_j in Eq. (F17) into the left and the right side, respectively, the following equation is obtained.

$$-1_1\cdot[n\cdot 2^{-m}]\leqq X<(1_2+1)\cdot[n\cdot 2^{-m}]$$

Substituting Eqs. (F30) and (F31) into the above equation,

$$-2^m[n\cdot 2^{-m}]\leqq X<2^m\cdot[n\cdot 2^{-m}] \quad (F140)$$

From Eq. (F1), $[n\cdot 2^{-m}]<2^{k+1}$ holds: substituting it into the above equation,

$$-1<X\cdot 2^{-(k+w+1)}<1 \quad (F141)$$

Next, Y is defined in the following manner.

$$Y=\left[\frac{X}{[n\cdot 2^{-m}]}\right]-[X\cdot v\cdot 2^{-(k+w+1)}] \quad (F142)$$

$$Y=\left[\frac{X}{[n\cdot 2^{-m}]}-X\cdot v\cdot 2^{-(k+w+1)}\right]+\delta_1 \quad (F143)$$

$\delta_1=0$ or 1

Substituting v from Eq. (F33),

$$Y=\left[X\cdot 2^{-(k+w+1)}\left\{\frac{\chi^{(k+w+1)}}{[n\cdot 2^{-m}]}-\left[\frac{\chi^{(k+w+1)}}{[n\cdot 2^{-m}]}\right]\right\}\right]+\delta_1 \quad (F144)$$

Setting $\epsilon_1=\frac{\chi^{(k+w+1)}}{[n\cdot 2^{-m}]}-\left[\frac{\chi^{(k+w+1)}}{[n\cdot 2^{-m}]}\right]$, $0\leqq\epsilon_1<1$.

so that the following equation is obtained taking Eq. (F141) into account.

$$Y=\begin{cases}\delta_1 & \text{for } X\geqq 0\\ \delta_1-1 & \text{for } X<0\end{cases} \quad (F144)$$

Setting $\delta_1=1-\delta_f$ and substituting Y,

$$\left[\frac{X}{[n\cdot 2^{-m}]}\right]-[X\cdot v\cdot 2^{-(k+v+1)}]=\begin{cases}1-\delta_j^* \text{ for } X\geq 0\\ -\delta_j^* \text{ for } X<0\end{cases} \quad \text{(F145)}$$

$$\therefore \left[\frac{X}{[n\cdot 2^{-m}]}\right]+\delta_j^*=\begin{cases}[X\cdot v\cdot 2^{-(k+v+1)}]+1 \text{ for } X\geq 0\\ [X\cdot v\cdot 2^{-(k+v+1)}] \text{ for } X<0\end{cases}$$

$\delta_j^*=0$, 1 and I_j in Eq. (F16) to obtain Eq. (F32). Next, Eq. (F19)" is verified. Eq. (F75) is applied to Eqs. (F89) and (F92) to substitute thereinto Eq. (F18)".

$$[(R_j+\delta_j^*\cdot n)2^{-m}]-\gamma_3+S+\alpha_j-\omega_{-m-1}+P\geq 0 \quad \text{(F112)"}$$

where $\gamma_3=0$, 1, 2

$$P=\begin{cases}P_1 \text{ for } I_j<0\\ P_2 \text{ for } I_j\geq 0\end{cases}$$

Thereafter, the following equation is obtained in the same manner as in the case of Eq. (F115)

$$-(S+A+(I_2+1))+\omega_{-m-1}\leq[(R_j+\delta_j^*\cdot n)2^{-m}]$$

$$\therefore -(S+A+(I_2+1))\cdot 2^m+\omega\cdot\delta_{(j-1)\lambda}[M_1\cdot 2^{-m}]\cdot 2^{+m}-\delta_j^*\cdot n\leq R_j$$

The left side of the above equation becomes minimum when $\delta_j^*=1$, by which the left of Eq. (F19)" is verified. Next, Eq. (F75) is applied to Eqs. (F90) and (F93) to substitute thereinto Eq. (F18)".

$$[(R_j-n+\delta_j^*\cdot n)2^{-m}]-\gamma_3+S+\alpha_j-\omega_{-m-1}+P<0 \quad \text{(F116)"}$$

wherein $\gamma_3=0$, 1, 2

$$P=\begin{cases}P_1 \text{ for } I_j<0\\ P_2 \text{ for } I_j\geq 0\end{cases}$$

Thereafter, the following equation is obtained in the same manner as in the case of obtaining Eq. (F119).

$$R_j<n-\delta_j^*\cdot n+(2-S+I_1)\cdot 2^m+\omega\cdot\delta_{(j-1)\lambda}[M_1\cdot 2^{-m}]2^m$$

The right side of the above equation becomes maximum when $\delta_j^*=0$, by which the right side of (F19)".

Next, it is verified that Eq. (F8) is necessary. Set R_L'=(the lower limit value of R_j)−(the lower limit value of R_{j+1}), then

$$R'_L=(-S\cdot 2^m+t_1\cdot n-\omega\cdot 2^m-(I_2+1)\cdot 2^m-n)+A\cdot 2^m\times(n\cdot 2^{-m-k}-1)+2^m\{[\omega\cdot\delta_\Lambda M_1\cdot 2^{-m}]-(\omega\cdot\delta_\Lambda M_1\cdot 2^{-m})+\omega\}$$

Thus, evidence has been given that Eq. (F8)" is necessary. The upper limit value remains unchanged.

Next, an operation

$$\sum_{j=1}^{\lambda} 2^{(j-1)\lambda}$$

is performed on both sides of Eq. (F18)" to obtain the following equation.

$$R_1=M_1\times M_2-n\times Q_1' \quad \text{(F123)'}$$

where

$$Q_1'=\sum_{j=1}^{\lambda}(I_j+\delta_j^*)2^{(j-1)\lambda} \quad \text{(F124)'}$$

It is evident that Eqs. (F21)" and (F22)" holds in the same manner as Eqs. (F125) to (F131).

Verification of Corollary 3

The conditions for which Eqs. (F35) to (F36) hold means that Eqs. (F90), (F92) and (F93) hold regardless of whether I_j is positive or negative and when $P_1=0$, $P_2=0$.

Accordingly, Eq. (F100) becomes as follows: PS

$$2^k-(1+\omega)+S\leq U \quad \text{(F100)'}$$

However, $0\leq U$ unconditionally holds from Eq. (F7)'. Further, it is apparent that when $I_j<-2^\lambda t_1-2$, Eqs. (F90) and (F93) do not hold.

$$"-I_1\leq I_j \quad \text{(F102)'}$$

Similarly, Eq. (F109) becomes as follows:

$$V<-2^{k+1}+1+2\omega+S+A \quad \text{(F109)'}$$

However, $V<0$ unconditionally holds from Eq. (F7)'. It is apparent that when $I_j>2^{\lambda+1}+2^\lambda t_2$, Eqs. (F89) and (F92) do not hold.

$$\therefore I_j\leq I_2 \quad \text{(F111)'}$$

Next, since $P=0$ holds in Eq. (F114), Eq. (F19)' holds by checking the proof of the theorem following Eq. (F112) and it is evident that Eq. (F8)' is a precondition of theorem.

Verification of Corollary 4

(A) Lower Limit Value of I_j

$I_j=-2^\lambda t_1-1$ is substituted into Eq. (F18).

$$R_j=2^\lambda R_{j+1}+M_1\cdot M_{2,j}-\omega\cdot\delta_\Lambda 2^\lambda M_1+\omega\cdot\delta_{(j-1)\lambda}M_1+(2^\lambda t_1+1)n$$

From Eq. (F12), $M_{2,j}\leq 0$, substituting it into the above equation,

$$R_j\geq 2^\lambda\{R_{j+1}-(-A\cdot v\cdot 2^{-k}-t_1\cdot n+\omega\cdot\delta_\Lambda\cdot M_1)\}+n(1-2^{-(k-\lambda-\log 2M)})+\omega\cdot M_1\cdot\delta_{(j-1)\lambda} \quad \text{(F146)}$$

Therefore, from Eqs. (F15) and (F3),

$$\omega\cdot\delta_{(j-1)\lambda}\cdot M_1\leq R_j \quad \text{(F147)}$$

Next, consider the case where $I_j=-2^\lambda t_1-2$. In this case, Eq. (F19) holds naturally. At this time, when obtaining R_j from Eq. (F18) with $I_j=-2^\lambda t_1-1$, R_j is smaller by $-n$ than in the case where $I_j=-2^\lambda t_1-2$, but the lower limit value of R_j is defined by Eq. (F147). Accordingly, it is seen that the lower limit value of R_j is

larger than the lower limit value of R_{j+1} shown by Eq. (F15) because $-A\cdot n2^{-k}-t_1\cdot n\leqq 0$ holds as the precondition for corollary 4.

(B) Upper Limit Value of I_j

$J_j=2^{\lambda+1}+2^{\lambda}t_2-2$ is subtituted to Eq. (F18). Setting $2^{\lambda+1}=2^{\lambda}+2^{\lambda}$,

$$R_j=2^{\lambda}\cdot R_{j+1}+M_1\cdot M_{2j}-\omega\cdot\delta_{j\lambda}M_1\cdot 2^{\lambda}+\omega\cdot M_1\cdot\delta_{(j-1)\lambda}-(2^{\lambda}+2^{\lambda}+2^{\lambda}t_2-2)n$$

From Eq. (F12), $M_{2j}\leqq 2^{\lambda}-1$; substituting it into the above equation,

$$R_j\leqq 2^{\lambda}\{R_{j+1}-(n+t_2\cdot n+\omega\cdot\delta_{j\lambda}M_1)\}+(M_1-n)(2^{\lambda}-1)+n+\omega\cdot M_1\cdot\delta_{(j-1)\lambda} \quad (F148)$$

Accordingly, from Eq. (F15)

$$R_j<n+\omega\cdot M_1\cdot\delta_{(j-1)\lambda} \quad (F149)$$

Next, consider the case where $2^{\lambda+1}+2^{\lambda}\cdot t_1\geqq I_j\geqq 2^{\lambda+1}+2^{\lambda}t_1-1$ holds. In this case, Eq. (F19) naturally holds. At this time, obtained from Eq. (F18) setting $I_j=2^{\lambda+1}+2^{\lambda}\cdot t_1-2$, R_j is found to be larger than the value $(2^{\lambda+1}+2^{\lambda}t_1\geqq I_j\geqq 2^{\lambda}\cdot t_1-1)$ of the first I_j by $+n$ or $+2n$, but the upper limit value of R_j at that time is defined by Eq. (F149).

Accordingly, it is seen that the upper limit value of R_j is smaller than the upper limit value of R_{j+1} because $t_2\cdot n\geqq 0$ holds as the precondition for the corollary 4.

Thus, evidence has been given that the corollary 4 holds.

Verification of Corollary 5.

It is evident that Eq. (F18) that Eq. (F19) holds, permitting the corollary 5 to hold.

As described above, according to the present invention, since $(M_1\times M_2)\div n$ can be executed by performing the multiplication and the division in parallel using the same clock, the quotient Q or/and the remainder R can be obtained at high speed.

$M_1\cdot M_{2j}$ CALCULATOR

Concerning the multiplication described in the theorem, supplemental explanation will be made below with respect to the condition $\omega=1$.

In Eq. (F10), setting $\omega=1$, the following equation is obtained.

$$M_{2j}'=M_{2j}-\delta_{j\lambda}\cdot 2^{\lambda}+\delta_{(j-1)\lambda}$$

A description will be given of the case where $\lambda=6$. M_{2ja}, M_{2jb} and M_{2jc} are defined as follows:

$$M_{2ja}=-\delta_{4(j-1)+4}\cdot 2^6+\delta_{4(j-1)+3}\cdot 2^5+2\cdot\delta_{4(j-1)+2}\cdot 2^4$$

$$M_{2jb}=-\delta_{4(j-1)+4}\cdot 2^4+\delta_{4(j-1)+3}\cdot 2^3+2\cdot\delta_{4(j-1)+2}\cdot 2^2$$

$$M_{2jc}=-\delta_{4(j-1)+2}\cdot 2^2+\delta_{4(j-1)+1}\cdot 2^1+2\cdot\delta_{4(j-1)+0}\cdot 2^0$$

Then, M_{2j} is as follows:

$$M_{2j}=M_{2ja}+M_{2jb}+M_{2jc}$$

M_{2ja}, M_{2jb} and M_{2jc} can be implemented by a circuit similar to that for Q_{ja}, Q_{jb} and Q_{jc} described previously in connection with the $-Q_j n$ calculator with reference to FIG. 62. With such an arrangement, the quantity of data representing $M_1\cdot M_{2j}'$ is decreased, permitting reduction of the circuit scale of the carry save adder 16.

Supplemental Description of Theorem 3

A description will be given of the general arrangement of a circuit for calculating the value of I_j by Eqs. (F35) and (F36). Since this circuit arrangement is identical with that of the circuit for calculating the value of Q_j by Eqs. (H2) and (H3) described previously in Example 2, the latter will hereinafter be described with reference to FIG. 66.

FIG. 66 illustrates a quotient calculator 9'' for calculating the value of Q_j based on Eqs. (H2) and (H3). Input signal lines 601, 602, 603 and 604 input therefrom variables R_{j+1}, δ_j, M_1, and n, respectively. In an AND circuit 871 is obtained $[\delta_j M_1\cdot 2^{-m}]$. An adder 620 performs an operation $[2R_{j+1}\cdot 2^{-m}]+[\delta_j M_1\cdot 2^{-m}]+2$. The last term $+2$ of this equation is generated within the adder 620. Circuits 621, 622 and 623 input therein n and output therefrom $[n\cdot 2^{-m}]$, $[-n\cdot 2^{-m}]$ and $[-2n\cdot 2^{-m}]$, respectively. The output of the adder 620 and the outputs of the circuits 621, 622 and 623 are added by adders 625, 627 and 628, respectively, and the output of the adder 620 is added with 0 in an adder 626. The adders 625 to 628 each output therefrom a 0 or 1, based on the following calculation, depending on whether the sign of a value $Q_j=0, 1, 2, 3$ is positive or 0, or negative.

$$[2R_{j+1}\cdot 2^{-m}]+[\delta_j M_1\cdot 2^{-m}]+[(1-Q_j)\cdot n\cdot 2^{-m}]+2$$

The output signs are indicated by signals QA1, QA2, QA3 and QA4, respectively. These signals QA1, QA2, QA3 and QA4 are applied to a circuit 629, from which a signal QB is provided based on logic shown in FIG. 67. The signal QB is equal to the value Q_j which satisfies Eqs. (H2) and (H3). In this way, the value Q_j can be output, as the signal QB, which satisfies Eqs. (H2) and (H3).

ANOTHER METHOD OF MULTIPLICATION-DIVISION

The following will describe that the calculation for the RSA cryptography can be performed even if the multiplier-divider which is a main constituent of the cryptosystem of the present invention is replaced with another kind of multiplier-divider. A description will be given first of another method of multiplication-division, then a multiplier-divider based on the calculation method and finally the arrangement of the cryptosystem.

ANOTHER METHOD OF MULTIPLICATION-DIVISION

This multiplication-division is performed by a method which can easily be deduced from an ordinary calculation method. At first, the multiplication $M_1\times M_2$ is executed and then the division $(M_1\times M_2)\div n$ is performed to obtain the remainder.

(A) Multiplication

The multiplication $M_1\times M_2$ is performed in the following manner. Let it be assumed that Z is a variable.

Step 1: $Z=0$

Step 2: The following operations are performed in an order $j=1, 2, \ldots l$.

$$Z=Z+M_1\times M_{2j}$$

Step 3: Halt.

(B) Division

A division $Z \div n$ is performed in the following manner. Here, R_j is a variable, and Z is represented as a binary number and divided equally into 2 l every λ bits and set as Z_j.

$$Z = \sum_{j=1}^{2l} Z_j \cdot 2^{(j-1)\lambda}$$

Step 4:

$$R_{j+1} = \sum_{j=l+1}^{2l} Z_j \cdot 2^{(j-l-1)\lambda}$$

Step 5: The following operations are executed in an order $j=l, l-1, \ldots 1$.

$$Q_j = \left[\frac{2^{\lambda} \cdot R_{j+1} + Z_j}{n} \right]$$

$$\cdot R_j = 2^{\lambda} \cdot R_{j+1} + Z_j - Q_j \cdot n$$

Step 6: $R = R_1$, halt. By steps 1 to 6, the remainder R of $(M_1 \times M_2) \div n$ can be obtained.

Here, the range of R_{j+1} in step 4 satisfies the following condition.

$$0 \leqq R_{j+1} < n$$

This reason is verified by the following based on conditions $0 \leqq M_1 < n$ and $0 \leqq M_2 < n$.

$$Z = R_{j+1} \cdot 2^{l\lambda} + \sum_{j=1}^{l} Z_j \cdot 2^{(j-1)\lambda}$$

$$Z = M_1 \times M_2 < n^2$$

Therefore, $0 \leqq R_{j+1} \cdot 2^{l\lambda} < n^2$ and

$$2^{l\lambda} < n < 2^{\lambda}$$

$$\therefore 0 \leqq R_{j+1} < n$$

For the verification, both sides of the equation of R_j are multiplied by $2^{(j-1)\lambda}$ and an addition of

$$\sum_{j=1}^{l}$$

is performed in respect of the multiplication result.

APPROXIMATION OF CALCULATION METHOD OF QUOTIENT Q_j IN DIVISION

The quotient Q_j can easily be obtained by a close approximation which involves omitting m bits from the variable R_{j+1} as is the case with the calculation of the quotient I_j in the aforementioned simultaneous multiplication-division and by a close approximation which involves multiplication using a reciprocal of the divisor in the division. That is, the division is performed using Q_j' defined by the following equation, instead of Q_j.

$$Q_j' = \begin{cases} [X_j \times v \times 2^{-m}] + 1 & \text{for } X_j \geqq 0 \\ [X_j \times v \times 2^{-m}] & \text{for } X_j < 0 \end{cases}$$

-continued

where

$$X_j = \sum_{i=0}^{l} [(2^{\lambda} \cdot R_{j+1,i}) \cdot 2^{-m}] + S$$

$$v = \left[\frac{2^u}{[n \cdot 2^{-m}]} \right]$$

L = the effective length of $n (2^{L-1} < n < 2^L)$

Here, m, S and u are defined as follows:

$$m \leqq L - \lambda - 6$$

$$2^{\lambda+2} + 2 \leqq S \leqq 2^{\lambda+3}$$

$$u \geqq L - m + \lambda + 2$$

In this case, it is the same as in the afore-described method of obtaining I_j by approximation that the following holds:

$$Q_j' = Q_j + \gamma_j, \ \gamma_j = 0, 1, \text{ or } 2$$

Here, R_j is divided into $R_{j,1}$ and $R_{j,0}$, and

$$R_j = \sum_{i=0}^{l} R_{j,i}$$

is set.

(C) Division Using Q_j'

In the case of using Q_j' in place of Q_j, the afore-described division changes as follows:

Step 4':

$$\sum_{i=0}^{l} R_{j+1,i} = \sum_{j=l+1}^{2l} Z_j \cdot 2^{(j-l-1)\lambda}$$

Step 5': The following is executed in the order $j=l, l-1, \ldots 1$.

$$Q_j' = \begin{cases} [X_j \times v \times 2^{-m}] + 1 & \text{for } X_j \geqq 0 \\ [X_j \times v \times 2^{-m}] & \text{for } X_j < 0 \end{cases}$$

$$\sum_{i=0}^{l} R_{j,i} = \sum_{i=0}^{l} 2^{\lambda} \cdot R_{j+1,i} + Z_j - Q_j' \cdot n$$

Step 6':

$$R_1 = \sum_{i=0}^{l} R_{1,i}$$

Step 7': If $R_1 \leqq 0$, go to step 9'

Step 8': $R_1 = R_1 + n$, go back to step 7'

Step 9': $R = R_1$ Halt

The above-described method of multiplication-division is called a "multiplication-division successive approximating calculation method".

(D) Multiplier-Divider

FIG. 68 illustrates the general arrangement of the multiplier-divider based on the simultaneous multiplication-division operating method described previously with regard to FIGS. 2 and 22 and Eqs. (3) to (24). A main adder 110_X is an assembly of the main adders 110_1 to 110_8 shown in FIG. 36, and a register 105_X is an assembly of the registers 105_1 to 105_8 shown in FIG. 34 and it has the function of shifting its content to left by

steps of four bits. An $M_1 \cdot M_{2j}$ calculator 140_X is an assembly of the calculators 140_1 to 140_8 shown in FIG. 37, and a $-Q \cdot n$ calculator 150_X is an assembly of the calculators 150_1 to 150_8 shown in FIG. 38. An adder 160_X is an assembly of the adders 160_1 to 160_8 depicted in FIG. 39, and an adding register 170_{LX} is an assembly of the registers 170_{L1} to 170_{L8} shown in FIG. 40. An adding register 170_{LY} is also a similar assembly of individual adding registers. Selectors 311_X and 312_X are assemblies of eight selectors 311 and 312 shown in FIG. 61, respectively. An adder 180_X is an assembly of the adders 180_1 to 180_8 shown in FIG. 4.

FIG. 69 illustrates a multiplier-divider based on the multiplication-division by successive approximation. In FIG. 69 the parts corresponding to those in FIG. 68 are identified by the same reference numerals. A selector 410 selects one of output signal lines of the calculators 140_X and 150_X and provides an output to a carry save adder 160_Y. Switching control of the selector 410 is effected by a signal on a control line 415. The carry save adder 160_Y is identical in construction with the adder 160_X. A register 105_Y is similar to the register 105_X but largely differs therefrom in that its content is shifted to the right by steps of four bits. Register sections 170_{LY} and 170_{RY} are registers of 1024-bit length. As depicted in FIG. 70, the register sections 170_{LY} and 170_{RY} are each formed by a series connection of a 514-bit register 419 and a 510-bit register 420 to constitute a 1024-bit register as a whole. This register has the function of shifting its content to the right and left by steps of four bit ($\lambda=4$). The register 170_Y has connected thereto a signal line 421 for determining the direction of shift, a shift command pulse input signal line 422, a signal line 423 for setting 0 in the content of the register, a register input signal line 425 and a register output signal line 426.

In the arrangement of FIG. 69, the calculation for obtaining the remainder of $(M_1 \times M_2) \div n$ is performed as follows: At first, the input signal line 411 of the selector 410 is selected, and the register sections 170_{LY} and 170_{RY} store 0 first and perform the multiplication by the aforesaid method utilizing the function of right shift by steps of four bits, thereby obtaining the value of $M_1 \times M_2$ on the register sections 170_{LY} and 170_{RY} of 1024-bit length. ($M_1 \times M_2$ is represented as the sum of numbers stored in the register sections 170_{LY} and 170_{RY}.)

Next, the input signal line 412 of the selector 410 is selected and the division is carried out by the quotient calculators 60 and 61 in the aforementioned manner utilizing the left shift function of the registers 170_{LY} and 170_{RY}. In this way, the remainder of the multiplication-division $(M_1 \times M_2) \div n$ can be obtained.

FIG. 70 illustrates the construction of a register 170_Y comprising the register sections 170_{LY} and 170_{RY}.

FIG. 71 illustrates the general arrangement of an embodiment of the cryptosystem of the present invention which employs the simultaneous multiplication-division method, and FIG. 72 shows the general arrangement of another embodiment of the present invention which employs the successive approximating multiplication-division method. In FIGS. 71 and 72, respective input and output signal lines correspond to those in the afore-described drawings and shown at the same positions. The register 420 in FIG. 22 corresponds to 510-bit-long register 420 in FIG. 70 which has the function of shifting its content to the right and left by steps of four bits, and the signal line 21' is a multiplication control signal line. As described previously in connection with FIG. 69, the register 105_Y is one that has the function of shifting its content to the right by steps of four bits. The multiplication control signal line 21' is led out from the slice section 25_8 and the direction of the signal on this line is opposite to that on the signal line in FIG. 71. The register 419 in each of the register sections 170_{LY} and 170_{RY} serves as a register equivalent to an assembly of the register 104_1 to 104_8 shown in FIG. 33. Because of such an arrangement, the calculation $C=M^e$ mod n can be performed using the variables e, n and M.

In the simultaneous multiplication-division method, it is also possible to calculate first $M_1 \times M_{2j}$ for each j and then perform the operation $-Q_j \times n$. In this case, as shown in FIG. 69, a selector is provided between the calculators 140_X and 150_X and the adder 160_X in the arrangement of FIG. 68, and $M_1 \times M_{2j}$ and $-Q_j \times n$ are alternately supplied from the selector to the adder 160_X for each j.

Furthermore, in the quotient calculating unit 9, the operation

$$Q_j = \left[\frac{M_1 \times M_{2j} + 2^{\lambda} R_{j+1}}{n} \right]$$

may be directly performed without using close approximation.

Although in the foregoing embodiments the quotient calculating unit 9 is provided independently of the sliceable section 25', it is also possible to provide quotient calculators 9_1 to 9_8 in the sliced sections 25_1 to 25_8, respectively, as shown in FIG. 73, for example, and to actuate only the quotient calculator 9_1 for the calculation for cryptography, holding the others inoperative. With such an arrangement, the cryptosystem of the present invention can be formed by eight LSI chips of the same configuration and any separate LSI chips need not be provided for the quotient calculating unit. Also it is possible to constitute LSIs including one part of the quotient calculating unit 9, for instance, the post-processing section 61 or pre-processing section 60, in the respective sliced sections 25_1 to 25_8, though not shown.

Conversely, since only one controller 8_1 in the sliced section 25_1 shown in FIG. 6 is made operative, it is possible to remove all the controllers 8_1 to 8_8 from the respective sliced sections 25_1 to 25_8, and provide a single controller of an LSI chip for controlling the sliced sections 8_1 to 8_8 and the quotient calculator 9.

As has been described in the foregoing, according to the present invention, the cryptosystem for implementing the RSA cryptograph $C=M^3$ mod n can easily be constituted through utilization of the present-day LSI technology even if the value of n is extremely large. For instance, the RSA cryptography employs the value $n = 10^{100}$ to 10^{200} and, in this case, the circuit scale of the cryptosystem is as large as 100K to 200K gates. According to the present invention, the cryptosystem can be formed by a small-scale ROM and 10-to-30K-gate LSI chips of the same configuration.

Furthermore, as will be appreciated from the foregoing, the value L-m is independent of the value L. Accordingly, the calculation by the quotient calculation post-processing section is independent of the value L, that is, the number of digits of the value n; therefore, the multiplication-division $R=M_1 \times M_2$ mod n and the operation $C=M^e$ and mod n can be performed increasing or decreasing the number of sliced sections. In other

words, the lengths of the encryption keys (n and e) can easily be changed by increasing or decreasing the number of sliced sections.

Besides, according to the present invention, the operation speed can be increased by the simultaneous multiplication-division as described previously. In this case, the main adding unit need not always be divided, that is, the arrangement shown in FIG. 68 may be employed.

It will be apparent that many modifications and variations may be effected without departing from the scope of the novel concepts of the present invention.

I claim:

1. An encryptosystem in which integers M, e and n ($0 \leqq M < n$) are applied to M-, e- and n-registers; variables C and M_2 are stored in C- and M_2-registers; the integer e being represented by

$$e \sum_{i=0}^{k} e_i \cdot 2^i$$

($e_i = 1$ or 0);

the variable C is initially set to 1; repetitive calculations are performed in accordance with the following Steps (1) and (2) for each value i in the order $i = k, k-1, k-2, \ldots 1, 0$; in Step (1) an operation $C = M_1 \times M_2 \bmod n$ is performed with $M_1 = C$ and $M_2 = C$; in Step (2) the value of e_i is checked and if $e_i = 1$, the operation $C = M_1 \times M_2 \bmod n$ is further performed with $M_1 = C$ and $M_2 = M$; and said repetitive calculations are completed with $i = 0$, producing the last C in the form of $C = M^e \bmod n$;

wherein a quotient calculating unit, a main adding unit and a controller are provided for performing the operation $C = M_1 \times M_2 \bmod n$, said main adding unit having an adding register for storing a variable R_j;

wherein, in order to perform the following operation in the order $j = l, l-1, l-2, \ldots 1$, thereby to obtain the last R_1 in the form of $C = M_1 \times M_2 \bmod n$:

$$R_j = M_1 \times M'_{2j} + 2^{\lambda} R_{j+1} - \left[\frac{M_1 \times M'_{2j} + 2^{\lambda} R_{j+1}}{n} \right] \times n$$

where

$$M'_{2j} = M_{2j} - \omega \delta_{j\lambda} \cdot 2^{\lambda} + \omega \delta_{(j-1)\lambda},$$

$$M_{2j} = \sum_{i=0}^{\lambda-1} \delta_{(j-1)\lambda+i} \cdot 2^i,$$

$$M_2 = \sum_{i=0}^{D-1} \delta_i \cdot 2^i, \ \delta_i = 0 \text{ or } 1,$$

$$R_{l+1} = 0 \text{ and } \omega = 0 \text{ or } 1,$$

where [ ] is a Gaussian symbol, [x] the largest possible integer smaller than or equal to x, and λ and l constants, said quotient calculating unit is connected to said C-, M_2- and n-registers and said main adding unit and performs an operation

$$\left[\frac{M_1 \times M'_{2j} + 2^{\lambda} R_{j+1}}{n} \right] = Q_j,$$

said main adding unit is connected to said quotient calculating unit and said C-, M_2- and n-registers and forms an operation $M_1 \times M_{2j} + 2^{\lambda} R_{j+1} - Q_j n$, and said controller performs control for obtaining said C by the respective calculations of said quotient calculating unit and said main adding unit.

2. A cryptosystem according to claim 1 wherein first and second adding registers are provided as said adding register; said variable R_j is divided into $R_{j,0}$ and

$$R_{j,1} \ (R_j = \sum_{i=0}^{l} R_{j,i});$$

$R_{j,0}$ and $R_{j,1}$ are stored in said first and second adding registers; and said main adding unit performs the following operation:

$$M_1 \times M'_{2j} + \sum_{i=0}^{l} 2^{\lambda} \cdot R_{j+1,i} - Q_j \cdot n$$

3. A cryptosystem according to claim 2 wherein said quotient calculating unit comprises a pre-processing section and a post-processing section connected thereto, said pre-processing section being supplied with said n to calculate $[2^u \div [n \cdot 2^{-m}]] = v$ (m and u being constants) and said post-processing section being supplied with said R_{j+1}, M_1, M_{2j} and v to calculate Q_j'' by approximation setting

$$\sum_{i=0}^{l} [2^{\lambda} R_{j+1,i} 2^{-m}] + \sum_{i=0}^{\lambda-1} [M_1 \cdot \delta_{(j-1)\lambda+i} 2^i \cdot 2^{-m}] + S = X_j''$$

in the case of $\omega = 0$ and, in the case of $\omega = 1$, setting

$$\sum_{i=0}^{l} [2^{\lambda} \cdot R_{j+1,i} \cdot 2^{-m}] + \sum_{i=0}^{\frac{\lambda}{2}-1} [M_1 \cdot \{ -\delta_{(j-1)\lambda+2i+2} \cdot 2^{2i+2} + \delta_{(j-1)\lambda+2i+1} \cdot 2^{2i+1} + 2\delta_{(j-1)\lambda+2i} \cdot 2^{2i} \} \cdot 2^{-m}] + S = X_j''$$

and setting as said Q_j

$$Q''_j = [X''_j \times v \times 2^{-n}] + 1 \text{ for } X''_j \geqq 0$$

$$Q''_j = [X''_j \times v \times 2^{-n}] \text{ for } X''_j < 0$$

or

$$Q''_j = [X''_j \times v \times 2^{-n}] + 1 \text{ for } X''_j > 0$$

$$Q''_j = [X''_j \times v \times 2^{-n}] \text{ for } X''_j \leqq 0$$

(S being a constant and $Q_j'' = Q_j - \delta$ holding and δ being an integer), and wherein compensating calculation means is included for obtaining, from said R_1, $R_1 + \delta \cdot n$ which satisfies $0 \leqq R_1 + \delta \cdot n < n$.

4. A cryptosystem according to claim 3 wherein said pre-processing section is a memory which is read out using $[n \cdot 2^{-m}]$ as its address.

5. A cryptosystem according to claim 2 wherein $\lambda = 1$ and $\omega = 0$; said quotient calculating unit is means supplied with said M_1, δ_j, n and R_{j+1}, for obtaining an approximate value Q_j'' of Q_j such that the calculation result obtained by calculating $[2R_{j+1} \cdot 2^{-m}] + [\delta_j M_1 \cdot 2^{-m}] - [Q_j n \cdot 2^{-m}]$ while changing Q_j successively varies in sign with respect to a reference value; and compensating calculation means is included for obtaining, from said R_1, $R_1 + \delta \cdot n$ which satisfies $0 \geqq R_1 + \delta \cdot n < n$ (δ being an integer).

6. A cryptosystem according to claim 3 or 5 wherein said main adding unit comprises an $M_1 \cdot M_{2j}$ calculating

section for calculating $M_1 \times M_{2j}'$, a $-Q_j \cdot n$ calculating section for calculating $-Q_j'' \times n$, said first and second adding registers for storing variables $R_{j+1,0}$ and $R_{j+1,1}$, means for multiplying the contents $R_{j+1,0}$ and $R_{j+1,1}$ of said first and second adding registers by 2^λ, a carry save adder for adding together the resulting $2^\lambda \cdot R_{j+1,0}$ and $2^\lambda \cdot R_{j+1,1}$, the calculated $M_1 \times M_{2j}'$ and the calculated $-Q_j'' \times n$ and storing the addition result in said first and second adding registers, and a carry propagation adder for adding two outputs from said carry save adder and storing the addition result in said C-register; said compensating calculation means comprises a selector for selecting one of the contents of said first and second adding registers, $R_{j+1,0}$ and $R_{j+1,1}$, and said 2^λ—multiplied values, $2^\lambda \cdot R_{j+1,0}$ and $2^\lambda \cdot R_{j+1,1}$, for input to said carry save adder, means for making zero one of M_1 and M_{2j}' applied to said $M_1 \cdot M_{2j}'$ calculating section, means for setting $-Q_j$ applied to said $-Q_j'' \cdot n$ calculating section to $+1$, and means for selecting $R_{j+1,0}$ and $R_{j+1,1}$ by said selector at the time of compensating calculation, making one of M_1 and M_{2j}' to be zero and $-Q_j$ to be $+1$, and activating said carry save adder and said carry propagation adder to perform an operation $R_1 + \delta \cdot n$.

7. A cryptosystem according to claim 3 or 5 wherein said main adding unit comprises an $M_1 \cdot M_{2j}$ calculating section for calculating $M_1 \times M_{2j}'$, a $-Q_j \cdot n$ calculating section for calculating $-Q_j'' \times n$, said first and second adding registers for storing the variable $R_{j+1,i}$ (i—0, 1), a carry save adder for adding $2^\lambda \cdot R_{j+1,i}$ (i=0, 1) obtained by multiplying $R_{j+1,i}$ (i=0, 1) by 2^λ, the calculated $M_1 \cdot M_{2j}'$ and the calculated $-Q_j'' \cdot n$ and storing the addition result in said first and second adding registers, and a carry propagation adder for adding two outputs from said carry save adder and storing the addition result in said C-register; and said compensating calculation means comprises a first selector for selecting either the one output from said carry save adder or the content of said C-register, a second selector for selecting either the other output from said carry save adder or n applied to said $-Q_j \cdot n$ calculating section, and means for selecting the content of the C-register and the n by said first and second selector, respectively, during compensating calculation, and activating said carry propagation adder to perform an operation $R_1 + \delta \cdot n$.

8. A cryptosystem according to claim 2 wherein said main adding unit comprises an $M_1 \cdot M_{2j}$ calculating section for calculating $M_1 \times M_{2j}'$, a $-Q_j \cdot n$ calculating section for calculating $-Q_j'' \times n$, said first and second adding registers for storing the variable $R_{j+1,i}$ (i—1, 0) a carry save adder for adding $2^\lambda R_{j+1}$ obtained by multiplying $R_{j+1,i}$ by 2^λ, the calculation result of said $M_1 \cdot M_{2j}$ calculating section and the calculation result of said $-Q_j \cdot n$ calculating section and storing the addition result in said first and second adding registers, and a carry propagation adder for adding two outputs from said carry save adder and storing the addition result in said C-register.

9. A cryptosystem according to claim 8, further including a selector for selecting one of the calculation results of said $M_1 \cdot M_{2j}$ calculating section and said $-Q_j \cdot n$ calculating section and supplying the selected calculation result to said carry save adder, and means for adding the selected calculation result and said $2^\lambda \cdot R_{j+1}$ and adding the addition result, for each j, with the other calculation result selected by said selector.

10. A cryptosystem according to claim 2, 3 or 5 wherein said main adding unit is divided into a plurality of sliced sections of the same function; said M_1 and n are divided into every fixed width of their binary integers and sequentially applied to said sliced sections; said M_{2j}' and Q_j are applied to said sliced sections in common to them; said sliced sections each perform a calculation $R_j = M_1 \times M_{2j}' + 2^\lambda R_{j+1} - Q_j'' \times n$ for the M_1, n, Q_j, M_{2j}' and R_{j+1} applied to them; and said sliced sections are each connected to a higher-order one of them via a connection signal line for applying thereto one part of the result of said calculation.

11. A cryptosystem according to claim 10, wherein an adder for performing the addition in the calculation $R_j = M_1 \times M_{2j}' + 2^\lambda R_{j+1} - Q_j'' \times n$ in each sliced section is composed of a plurality of carry save adders; the number of bits of each of an input and an output signal line of said carry save adder is selected larger than the number of binary bits of the fixed width of said integers; and means is included for supplying the most significant one of binary bits of the fixed width of said integers on the low-order side in the output signal line of each carry save adder to a corresponding one of the carry save adders of the higher-order sliced sections via a part of said connection signal line, and for applying said most significant bit applied from the lower-order sliced sections to the corresponding carry save adder, for applying λ bits of the last stage output signal line of said carry save adder to the higher-order sliced section via the other part of said connection signal line to apply a signal of said λ bits from the lower-order sliced section to the last-stage output of said carry save adder.

12. A cryptosystem according to claim 10 wherein said sliced section each include an M_2-register for input therein said divided M_2.

13. A cryptosystem according to claim 12 wherein said sliced sections each include a selector controlled by said e_i to select one of said M and C for input to said M_2-register.

14. A cryptosystem according to claim 13 wherein said sliced section each includes an M-register for storing said divided M.

15. A cryptosystem according to claim 13 wherein said sliced sections each include an n-register for storing said divided n.

16. A cryptosystem according to claim 13 wherein said sliced section each include an e-register for storing said divided e.

17. A cryptosystem according to claim 13 wherein said sliced sections each include a C-register for storing said divided C.

18. A cryptosystem according to claim 13 wherein said sliced sections each include at least one part of said quotient calculating section, and only one of said quotient calculating sections of said sliced sections is made operable.

19. A cryptosystem according to claim 13 wherein said sliced sections each include said controller, and only one of said controllers of said sliced sections is made operable.

20. A cryptosystem in which integers M, e and n ($0 \leqq M < n$) are applied to M, e and n registers; variables C and M_2 are stored in C- and M_2-registers; the integer e being represented by

$$e = \sum_{i=0}^{k} e_i \cdot 2^i$$

(e_i=0 or 1); the variable C is initially set to 1; repetitive calculations are performed in accordance with the fol-

lowing Steps (1) and (2) for each value i in the order i=k, k−1, k−2, . . . 1, 0; in Step (1) an operation $C \equiv M_1 \times M_2 \bmod n$ is performed with $M_1=C$ and $M_2=C$; in Step (2), the value of e_i is checked and if $e_i=1$, the operation $C \equiv M_1 \times M_2 \bmod n$ is further performed with $M_1=C$ and $M_2=M$; and said repetitive calculations are completed with i=0, producing the last C in the form of $C \equiv M^e \bmod n$;

said cryptosystem comprising a main adding unit including at least an $M_1 \cdot M_{2j}$ calculating section for calculating $M_1 \times M_{2j}$, a $-Q_j \cdot n$ calculating section for calculating $-Q_j'' \times n$, a selector for selecting one of the calculation results $M_1 \cdot M_{2j}$ and $-Q_j'' \cdot n$, an adding register and an adder for adding the content of said adding register and the output of said selector and storing the addition result, in said adding register, a controller, and a quotient calculating unit;

wherein the main adding unit is controlled by said controller so that a 0 is applied as a variable Z to said adding register, said calculation result $M_1 \cdot M_{2j}$ is selected by said selector, an operation $Z=Z+M_1 \times M_{2j}$ is performed in the order j=1, 2, . . . l to obtain $M_1 \cdot M_2 \equiv Z$,

$$R_{l+1} = \sum_{j=l+1}^{2l} \cdot Z \cdot 2^{(j-l)\lambda} \text{ or } Z = \sum_{j=1}^{2l} \cdot Z_j \cdot 2^{(j-l)\lambda}$$

(λ being constant) is applied to said adding register, said calculation result $-Q_j'' \cdot n$ is selected by said selector, and an operation $R_j=2^{\lambda}R_{j+1}+Z_j-Q_j'' \cdot n$ is performed in the order j=l, l−1, . . . 1;

wherein said main adding unit is divided into a plurality of sliced sections of the same function, said M_1 and n are divided into every fixed width of their binary integers and sequentially applied to said sliced sections, said M_{2j} and Q'' are applied to said sliced sections in common to them, said sliced sections each perform said operations $Z=Z+M_1 \times M_{2j}$ and $R=2^{\lambda}R_{j+1}+Z_j-Q_j'' \cdot n$ for the M_1, n, Q_j'' and M_{2j} applied to them, said sliced sections are each connected to a higher-order one of them via a first connection signal line for applying thereto one part of the calculation result Z, and said sliced sections are each connected to a lower-order one of them via a second connection signal line for applying thereto the calculation result R_j.

21. A cryptosystem in which integers M, e and n ($0 \leq M < n$) are applied to M, e and n registers; variables C and M_2 are stored in C- and M_2-registers; the integer e being represented by

$$e = \sum_{i=0}^{k} \cdot e_i \cdot 2^i$$

(e_i=0 or 1); the variable C is initially set to 1; repetitive calculations are performed in accordance with the following Steps (1) and (2) for each value i in the order i=k, k−1, k−2, . . . 1, 0; in Step (1) an operation $C \equiv M_1 \times M_2 \bmod n$ is performed with $M_1=C$ and $M_2=C$; in Step (2), the value of e_i is checked and if $e_i=1$, the operation $C \equiv M_1 \times M_2 \bmod n$ is further performed with $M_1=C$ and $M_2=M$; and said repetitive calculations are completed with i=0, producing the last C in the form of $C \equiv M^e \bmod n$;

said cryptosystem comprising a main adding unit including at least an $M_1 \cdot M_{2j}$ calculating section for calculating $M_1 \times M_{2j}$, a $-Q_j \cdot n$ calculating section for calculating $-Q_j'' \times n$, a selector for selecting one of the calculation results $M_1 \cdot M_{2j}$ and $-Q_j'' \cdot n$, an adding register and an adder for adding the content of said adding register and the output of said selector and storing the addition result in said adding register, a controller, and a quotient calculating unit;

wherein a 0 is applied as a variable Z to said adding register, said calculation result $M_1 \cdot M_{2j}$ is selected by said selector, an operation $Z=Z+M_1 \times M_{2j}$ is performed in the order j=1, 2, . . . l to obtain $M_1 \cdot M_2 \equiv Z$, then R_{l+1} of

$$Z = \sum_{j=1}^{2l} \cdot Z_j \cdot 2^{(j-l)\lambda} \text{ (λ being a constant)}$$

$$R_{l+1} = \sum_{i=l+1}^{2l} \cdot Z_j \cdot 2^{(j-l-1)\lambda}$$

is applied to said adding register, said calculation result $-Q_j'' \cdot n$ is selected by said selector, said quotient calculating unit comprises a calculating section for calculating $X_j=[2^{\lambda} \cdot R_j 2^{-m}]+S$ (S being a constant) and a calculating section for calculating

$$\nu = \left[\frac{2^n}{[n \cdot 2^{-m}]}\right],$$

and said quotient calculating unit is controlled by said controller to calculate

$$Q_j''=[X_j \times \nu \times 2^{-n}]+1$$

when

$$X_j \geq 0$$

$$Q_j''=[X_j \times \nu \times 2^{-n}]$$

when

$$X_j < 0$$

or

$$Q_j''=[X_j \times \nu \times 2^{-n}]+1$$

when

$$X_j > 0$$

$$Q_j''=[X_j \times \nu \times 2^{-n}]$$

when

$$X_j \leq 0$$

and calculate $R_j=2^{\lambda} \cdot R_j+R_j-Q_j' \cdot n$ in the order j=l, l−1, . . . 1;

wherein compensation calculation means is included for calculating, when $R_1 \geq 0$, $R_1=R_1+n$ until $R_1 \geq 0$ is obtained;

wherein said main adding unit is divided into a plurality of sliced sections of the same function, said M_1 and n are applied to said sliced sections while being sequentially divided for each fixed width of their integers, said M_{2j} and Q'' are applied to said sliced sections in common to them, said sliced sections

each perform said operations $Z=Z'+M_1\times M_{2j}'$ and $R_j=2^{\lambda}R_{j+1}+Z_j-Q_j''\cdot n$ for the M_1, n, Q_j'' and M_{2j}' applied to them, said sliced sections are each connected to a higher-order one of them via a first connection signal line for applying thereto one part of the calculation result Z, and said sliced sections are each connected to a lower-order one of them via a second connection signal line for applying thereto the calculation result R_j.

22. A cryptosystem according to claim 20 wherein first and second adding registers are provided as said adding register; said variable R_j is divided into $R_{j,0}$ and

$$R_{j,1} \text{ (i.e. } R_j = \sum_{i=0}^{1} R_{j,i}\text{)};$$

$R_{j,0}$ and $R_{j,1}$ are stored in said first and second adding registers; and said main adding unit performs the following operation:

$$\sum_{i=0}^{1} R_{j,i} = \sum_{i=0}^{1} 2^{\lambda}\cdot R_{j+1,i} + Z_j - Q_j''\cdot n.$$

* * * * *